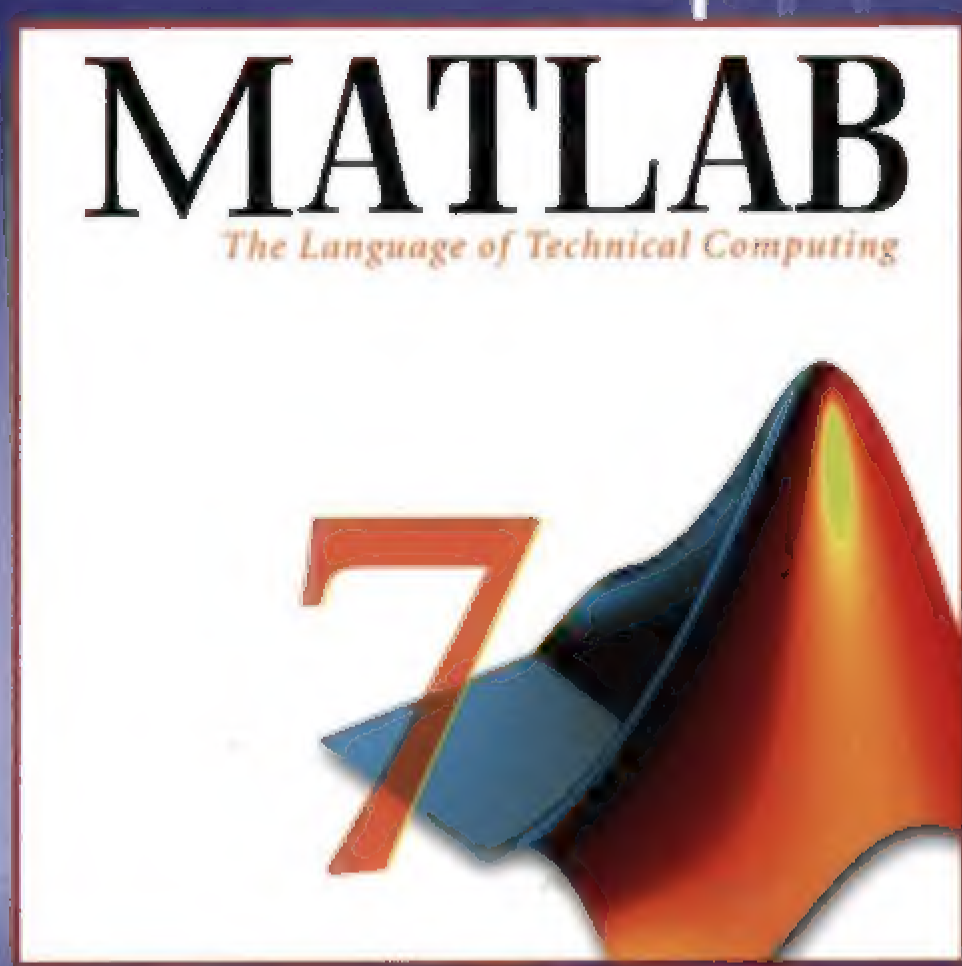
MATLAB有限元结构动力学分析与工程应用

徐斌　高跃飞　余龙　编著





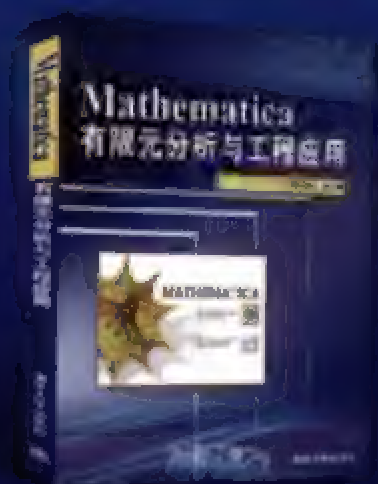
清华大学出版社

MATLAB有限元
结构动力学分析与工程应用













本书特色

- 采用MATLAB作为编程平台，利用MATLAB强大的科学计算和符号运算功能，帮助读者轻松跨越繁琐的公式推导和复杂的编程技巧，获得最佳的学习效率。
- 鉴于国内基于MATLAB的有限元分析介绍主要停留在静力学问题分析上，很少或较少篇幅涉及动力学分析，本书系统深入地介绍基于MATLAB的结构动力学分析。
- 本书除了介绍有限元的基本理论，还介绍了作者多年来基于MATLAB的工程仿真成果。

ISBN 978-7-302-21148-8



9 787302 211488 >
定价：37.00元

MATLAB 有限元结构动力学分析与工程应用

徐　斌　高跃飞　余　龙　编著



清华大学出版社
北　京

内 容 简 介

本书共 8 章，系统地阐述了基于有限元和 MATLAB 软件的结构动力学计算和它在工程数值仿真中的应用，包括有限元的基本方法和步骤、结构的动力特性和响应分析、单元的质量矩阵和刚度矩阵的建立及典型结构的动力学分析、工程应用和数值仿真等内容。

本书可作为力学、机械、航空航天、土木、水利等专业的本科生和研究生教材，也可作为上述专业教师、工程师和科研人员的参考用书。

图书在版编目(CIP)数据

MATLAB 有限元结构动力学分析与工程应用/徐斌，高跃飞，余龙编著. —北京：清华大学出版社，2009.12

ISBN 978-7-302-21148-8

Ⅰ. M…　Ⅱ. ①徐…　②高…　③余…　Ⅲ. 结构动力学—有限元分析：动力学分析—应用程序，MATLAB
Ⅳ. O342-39

中国版本图书馆 CIP 数据核字(2009)第 177073 号

责任编辑：刘天飞　桑任松
封面设计：杨玉兰
版式设计：北京东方人华科技有限公司
责任校对：周剑云
责任印制：王秀菊
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：清华大学印刷厂
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185×260　**印　张**：25　**字　数**：604 千字
版　　次：2009 年 12 月第 1 版　　**印　次**：2009 年 12 月第1 次印刷
印　　数：1～4000
定　　价：37.00 元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。联系电话：(010)62770177 转 3103　　产品编号：032422-01

前　言

有限元法发展至今天，已成为工程数值分析的有力工具，在理论和实践上均取得了令人瞩目的成就，事实上它已经发展成为工程领域中一门不可或缺的技术。本书采用在当今工程和教育界非常流行的数学软件 MATLAB 来进行有限元的分析和应用，特别是进行结构的动力学分析。

本书的一大特色是采用 MATLAB 作为编程平台，利用 MATLAB 强大的科学计算和符号运算功能，帮助读者轻松跨越繁琐的公式推导和复杂的编程技巧，获得最佳的学习效率。国内基于 MATLAB 的有限元分析介绍主要停留在静力学问题分析上，很少或较少篇幅涉及动力学分析，基于此，系统、深入地介绍基于 MATLAB 的结构动力学分析，是本书的主要特色之二。本书除了介绍有限元的基本理论，还将介绍作者多年来基于 MATLAB 的工程仿真成果，是本书的主要特色之三。

本书详细、系统地介绍基于 MATLAB 的结构动力学的基本分析，在写作上，采用理论和程序紧密结合的方法，以加强读者的感性认识，更好地理解有限元理论，每章后面都配有丰富和详细的工程仿真和应用实例，这也是诸多与有限元应用有关的本科生、研究生、科研人员和工程技术人员所希望得到的资料。本书不仅能让不懂此软件分析的读者入门，而且能让入门者进阶，最后达到精通，能让精通者应用到工程实际中，解决实际工程计算仿真和应用问题。

本书的内容共分 8 章和 1 个附录。第 1 章主要介绍有限元的基本方法和应用步骤；第 2 章主要讲述结构的动力特性和响应分析；第 3～7 章主要介绍各种有限元单元以及各种典型工程结构，包括各种单元的质量矩阵和刚度矩阵的建立以及基本的结构动力学分析(固有频率的求解和动响应分析)；第 8 章为工程应用和数值仿真部分，主要介绍基于 MATLAB 的结构动力学分析在结构领域的一些应用；附录针对 MATLAB 语言和其他高级编程语言的不同之处，对 MATLAB 在本书中使用到的功能进行简要的介绍。(另外，本书正文中用句点“.”表示一句话结束，含义与句号“。”相同。)

适用对象：本书内容专业，是一本难得的、系统的工程书籍，能够帮助读者更好地解决问题，可以作为在校大学生、研究生、教师、工程师和科研人员的参考手册，亦可作为广大工程技术人员的参考用书。

本书由徐斌(西北工业大学)、高跃飞(中北大学)和余龙(西北工业大学)等负责编写。第 2 章、第 6 章、第 8 章 8.1～8.3 节由徐斌编写，第 1 章 1.1 节和 1.2 节、第 4 章、第 5 章由高跃飞编写，第 1 章 1.3 节、第 3 章、第 7 章以及附录由余龙编写，第 8 章 8.5 节由杨永锋编写。全书的统稿及审校工作由徐斌负责。还要特别感谢家人在作者写作本书时所做的支持和理解。由于本书程序量大，为了方便读者学习，本书中的所有程序均已存在网站下载资源中。程序是按章分类的，每个文件名都有一个相应的序号，根据书中的模型名称，或者 M 文件名称直接在网站下载资源中查找相对应的文件名即可方便地找到。另外，书中所有程序都已上传至科研中国网站 http://www.sciei.com。如果有任何技术问题，

欢迎大家到科研论坛 http://bbs.sciei.com 进行交流，相信您能够得到满意的答复，同时也欢迎 MATLAB 爱好者来这里展现您的能力。

由于作者水平有限，时间仓促，书中不妥之处在所难免，敬请广大读者不吝指正！任何意见和建议均可通过电子邮箱 xubind@sina.com 发给作者。读者也可通过这个邮箱索取本书的源程序和数据文件。

徐　斌

于西北工业大学



目　录

第 1 章　有限元法基础

1.1　有限元法简介

在工程与科学的现代系统分析中，对复杂系统计算模型的建立进行了大量的研究，人们已经能够得到系统应遵循的基本方程和相应的定解条件. 这些方程一般为常微分方程或偏微分方程，只有少数问题能够用解析方法得到精确解，多数问题需要利用数值方法来求解. 有限元法(又称有限单元法)是近代发展起来的解决复杂结构问题的一种有效数值方法.

有限元法的基本思想是将连续的求解区域离散为一组有限个、按一定方式相互联结在一起的单元的组合体. 由于单元能按不同的联结方式进行组合，且单元本身又可以有不同的形状，因此可以模型化几何形状复杂的求解域. 有限元法作为数值分析方法的另一个重要特点是利用在每一个单元内假设的近似函数来分片地表示全求解域上待求的未知场函数. 单元内的近似函数通常由未知场函数或及其导数在单元的各个节点的数值和其插值函数来表达. 这样一来，一个问题的有限元分析中，未知场函数或其导数在各个节点上的数值就成为新的未知量(也即自由度)，从而使一个连续的无限自由度问题变成离散的有限自由度问题. 一经求解出这些未知量，就可以通过插值函数计算出各个单元内场函数的近似值，从而得到整个求解域上的近似解. 显然随着单元数目的增加，也即单元尺寸的缩小，或者随着单元自由度的增加及插值函数精度的提高，解的近似程度将不断改进. 如果单元是满足收敛要求的，近似解最后将收敛于精确解.

从确定单元特性和建立求解方程的理论基础和途径来说，早期提出有限元法时是利用直接刚度法，它来源于结构分析的刚度法. 1963—1964 年，有限元法被证明是基于变分原理的 Ritz(里兹)法的另一种形式，从而使 Ritz 法分析的所有理论基础都适用于有限元法，确认了有限元法是处理连续介质问题的一种普遍方法. 利用变分原理建立有限元方程和经典里兹法的主要区别是有限元法假设的近似函数不是在全求解域而是在单元上规定的，而且事先不要求满足任何边界条件，因此它可以用来处理很复杂的连续介质问题. 从 20 世纪 60 年代后期开始，利用加权余量法来确定单元特性和建立有限元求解方程的方法得到了普遍的应用. 有限元法中所利用的主要是 Galerkin(伽辽金)法，它可以用于已知问题的微分方程和边界条件，但是变分的泛函尚未找到或者根本不存在的情况，进一步扩大了有限元法的应用领域[1].

近年来，随着计算机技术的快速发展和各种商业化有限元软件的不断完善，有限元法逐渐成为动力学分析所普遍采用的一种有效方法.

1.2 建立有限元方程的基本方法

1.2.1 加权余量法

基于微分方程等效积分的加权余量法是求解微分方程近似解的一种有效方法，有限元法中可以用加权余量法来建立有限元方程. 工程中的多数分析问题是以未知场函数应满足的微分方程和边界条件的形式表示的，一般可表示为未知函数 $\boldsymbol{u}$ 应满足的微分方程组

$$\boldsymbol{A}(u)=\begin{Bmatrix} A_1(u) \\ A_2(u) \\ \vdots \end{Bmatrix}=\boldsymbol{0} \quad (在\ \Omega\ 内) \tag{1-1}$$

和应满足的边界条件

$$\boldsymbol{B}(u)=\begin{Bmatrix} B_1(u) \\ B_2(u) \\ \vdots \end{Bmatrix}=\boldsymbol{0} \quad (在\ \Gamma\ 上) \tag{1-2}$$

式中，域 Ω 可以是体积域、面积域等；而 Γ 是域 Ω 的边界. 上述的未知函数 $\boldsymbol{u}$ 可以是标量场(如温度)，也可以是向量场(如位移、应力、应变等). $\boldsymbol{A}$ 、 $\boldsymbol{B}$ 是对于独立变量的微分算子.

由于式(1-1)在域 Ω 中的任一点均必须为零，因而有

$$\int_\Omega \boldsymbol{V}^{\mathrm{T}}\boldsymbol{A}(u)\mathrm{d}\Omega=\int_V (v_1A_1(u)+v_2A_2(u)+\cdots)\mathrm{d}\Omega=0 \tag{1-3}$$

其中

$$\boldsymbol{V}=\begin{Bmatrix} v_1 \\ v_2 \\ \vdots \end{Bmatrix}$$

是函数向量，是一组与微分方程个数相等的任意函数.

式(1-3)是与微分方程组(1-1)完全等效的积分形式. 可以断言，若积分方程(1-3)对于任意的函数向量 $\boldsymbol{V}$ 成立，则微分方程组(1-1)必然在域 Ω 内任一点都满足. 这是因为假定微分方程组(1-1)在域 Ω 内某些点或一部分子域中不满足，相应地可找到适当的函数 $\boldsymbol{V}$ 使式(1-3)亦不等于零.

同理若对边界上任一点式(1-2)成立，则对于一组任意函数 $\bar{\boldsymbol{V}}$ 有

$$\int_\Gamma \bar{\boldsymbol{V}}^{\mathrm{T}}\boldsymbol{B}(u)\mathrm{d}\Gamma=\int_\Gamma (\bar{v}_1B_1(u)+\bar{v}_2B_2(u)+\cdots)\mathrm{d}\Gamma=0 \tag{1-4}$$

因此，积分方程

$$\int_\Omega \boldsymbol{V}^{\mathrm{T}}\boldsymbol{A}(u)\mathrm{d}\Omega+\int_\Gamma \bar{\boldsymbol{V}}^{\mathrm{T}}\boldsymbol{B}(u)\mathrm{d}\Gamma=0 \tag{1-5}$$

对于所有的 $\boldsymbol{V}$ 和 $\bar{\boldsymbol{V}}$ 成立等效于满足微分方程(1-1)和边界条件(1-2). 式(1-5)称为微分方程的等效积分形式，也称为等效积分的“强”形式. 在上述的讨论中，假定积分 $\int_\Omega \boldsymbol{V}^{\mathrm{T}}\boldsymbol{A}(u)\mathrm{d}\Omega$ 和 $\int_\Gamma \bar{\boldsymbol{V}}^{\mathrm{T}}\boldsymbol{B}(u)\mathrm{d}\Gamma$ 是可计算的，这就要求函数 $\boldsymbol{V}$ 和 $\bar{\boldsymbol{V}}$ 的选取必须满足可积的条件.

在很多情况下，对式(1-5)可进行分部积分，得到另一种等效积分形式

$$\int_{\Omega} \boldsymbol{C}^{\mathrm{T}}(v)\boldsymbol{D}(u)\mathrm{d}\Omega + \int_{\Gamma} \boldsymbol{E}^{\mathrm{T}}(\bar{v})\boldsymbol{F}(u)\mathrm{d}\Gamma = 0 \tag{1-6}$$

式中，$\boldsymbol{C}$、$\boldsymbol{D}$、$\boldsymbol{E}$和$\boldsymbol{F}$是微分算子. 式(1-6)称为微分方程等效积分的“弱”形式. 积分方程(1-6)中所包含的导数的阶数较式(1-5)中的$\boldsymbol{A}$低，这就对函数u的连续性要求降低，只需有较低阶的连续性就可以了. 在求解域Ω中，若场函数u是精确解，则在域Ω中任一点都满足微分方程(1-1)，同时在边界Γ上任一点都满足边界条件(1-2). 此时，等效积分形式的式(1-5)或式(1-6)也必然满足. 但是对于多数应用问题，这样的精确解是难以获得的，因而人们致力于寻找具有一定精度的近似解. 加权余量法是获取微分方程近似解的一种有效方法.

对于式(1-1)和式(1-2)所描述的问题，未知场函数u可用带有待定参数的近似函数来表示. 这种近似函数是一簇已知函数，一般可表示为以下形式

$$u \approx \tilde{u} = \sum_{i=1}^{n} N_i \alpha_i = N\alpha \tag{1-7}$$

式中，α_i是待定参数；N_i是称为试探函数的已知函数，可取自线性独立的完全函数序列. 此外，这种近似函数的选取应满足边界条件和连续性的要求.

一般在n取有限项数的情况下近似解不能精确满足微分方程(1-1)和边界条件(1-2)，将产生残差$\boldsymbol{R}$和$\bar{\boldsymbol{R}}$

$$\boldsymbol{A}(N\alpha) = \boldsymbol{R}, \quad \boldsymbol{B}(N\alpha) = \bar{\boldsymbol{R}} \tag{1-8}$$

这种残差称为余量.

对于式(1-5)，用n个规定的函数来代替任意函数$\boldsymbol{V}$和$\bar{\boldsymbol{V}}$

$$\boldsymbol{V} = \boldsymbol{W}_j \quad \bar{\boldsymbol{V}} = \bar{\boldsymbol{W}}_j \quad (j = 1,2,\cdots,n)$$

则可得到近似的等效积分形式

$$\int_{\Omega} \boldsymbol{W}_j^{\mathrm{T}} \boldsymbol{A}(N\boldsymbol{\alpha})\mathrm{d}\Omega + \int_{\Gamma} \bar{\boldsymbol{W}}_j^{\mathrm{T}} \boldsymbol{B}(N\boldsymbol{\alpha})\mathrm{d}\Gamma = 0 \ (j = 1,2,\cdots,n) \tag{1-9}$$

表示成余量的形式为

$$\int_{\Omega} \boldsymbol{W}_j^{\mathrm{T}} \boldsymbol{R}\mathrm{d}\Omega + \int_{\Gamma} \bar{\boldsymbol{W}}_j^{\mathrm{T}} \bar{\boldsymbol{R}}\mathrm{d}\Gamma = 0 \quad (j = 1,2,\cdots,n) \tag{1-10}$$

式(1-9)和式(1-10)说明通过选择待定参数$\boldsymbol{\alpha}$可使余量在某种平均意义上等于零. $\boldsymbol{W}_j$和$\bar{\boldsymbol{W}}_j$称为权函数.

令余量的加权积分为零得到一组方程，可用来求解近似函数的待定参数$\boldsymbol{\alpha}$，从而得到原问题的近似解. 展开式(1-9)，有

$$\begin{gathered}
\int_{\Omega} \boldsymbol{W}_1^{\mathrm{T}} \boldsymbol{A}(N\boldsymbol{\alpha})\mathrm{d}\Omega + \int_{\Gamma} \bar{\boldsymbol{W}}_1^{\mathrm{T}} \boldsymbol{B}(N\boldsymbol{\alpha})\mathrm{d}\Gamma = 0 \\
\int_{\Omega} \boldsymbol{W}_2^{\mathrm{T}} \boldsymbol{A}(N\boldsymbol{\alpha})\mathrm{d}\Omega + \int_{\Gamma} \bar{\boldsymbol{W}}_2^{\mathrm{T}} \boldsymbol{B}(N\boldsymbol{\alpha})\mathrm{d}\Gamma = 0 \\
\vdots \\
\int_{\Omega} \boldsymbol{W}_n^{\mathrm{T}} \boldsymbol{A}(N\boldsymbol{\alpha})\mathrm{d}\Omega + \int_{\Gamma} \bar{\boldsymbol{W}}_n^{\mathrm{T}} \boldsymbol{B}(N\boldsymbol{\alpha})\mathrm{d}\Gamma = 0
\end{gathered}$$

以上方程中若$\boldsymbol{A}$中的元素个数为m_1，边界条件$\boldsymbol{B}$中的元素个数为m_2，则权函数$\boldsymbol{W}_j$ $(j = 1,2,\cdots,n)$是m_1阶的函数列阵，$\bar{\boldsymbol{W}}_j$ $(j = 1,2,\cdots,n)$是m_2阶的函数列阵. 近似函数取的

项数n越多，近似解的精度越高．当项数n趋于无穷时，近似解将收敛于精确解．

对于等效积分“弱”形式，代入近似解的近似形式为

$$\int_{\Omega}\boldsymbol{C}^{\mathrm{T}}(\boldsymbol{W}_j)\boldsymbol{D}(N\alpha)\mathrm{d}\Omega+\int_{\Gamma}\boldsymbol{E}^{\mathrm{T}}(\boldsymbol{W}_j)\boldsymbol{F}(N\alpha)\mathrm{d}\Gamma=0 \quad (j=1,2,\cdots,n) \tag{1-11}$$

这种采用使余量积分为零来求解微分方程近似解的方法称为加权余量法．权函数可以从任何独立的完整函数集来选取．按照权函数选取的不同可以给出不同的加权余量方法．常见的权函数有以下几种(其他的方法可见参考文献[2])．

1．配点法

该方法用 Dirac δ函数作为权函数

$$\boldsymbol{W}_j=\bar{\boldsymbol{W}}_j=\delta(x-x_j) \tag{1-12}$$

若域Ω是独立坐标x的函数，则有：当$x\neq x_j$时$W_i=0$，且有

$$\int_{\Omega}\boldsymbol{W}_j\mathrm{d}\Omega=\boldsymbol{I} \quad (j=1,2,\cdots,n)$$

这种方法强迫余量在域Ω内的n个点上等于零．

2．最小二乘法

当近似函数取为

$$\tilde{u}=\sum_{i=1}^{n}N_i\alpha_i$$

权函数取为

$$\boldsymbol{W}_j=\frac{\partial \boldsymbol{R}}{\partial\alpha_j}=\frac{\partial}{\partial\alpha_j}\boldsymbol{A}\left(\sum_{i=1}^{n}N_i\alpha_i\right) \tag{1-13}$$

由

$$\int_{\Omega}\boldsymbol{W}_j^{\mathrm{T}}\boldsymbol{R}\mathrm{d}\Omega=0$$

有

$$\frac{\partial}{\partial\alpha_j}\int_{\Omega}\boldsymbol{R}^2\mathrm{d}\Omega=\frac{\partial}{\partial\alpha_j}\int_{\Omega}\boldsymbol{A}^2\left(\sum_{i=1}^{n}N_i\alpha_i\right)\mathrm{d}\Omega=0$$

其实质是使积分

$$J(\alpha_i)=\int_{\Omega}\boldsymbol{A}^2\left(\sum_{i=1}^{n}N_i\alpha_i\right)\mathrm{d}\Omega$$

取最小值．

3 Galerkin 法

在该方法中，用试探函数的序列作为权函数．取权函数为

$$\boldsymbol{W}_j=\frac{\partial\tilde{u}}{\partial\alpha_j},\quad \bar{\boldsymbol{W}}_j=-\frac{\partial\tilde{u}}{\partial\alpha_j} \tag{1-14}$$

或

$$\boldsymbol{W}_j=N_j,\quad \bar{\boldsymbol{W}}_j=-N_j \tag{1-15}$$

以上几种方法可以用下面的二阶常微分方程求解来说明.

【例1.1】 对于问题

$$\begin{cases} \dfrac{\partial^2 u}{\partial x^2} + u = -x & (0 \leqslant x \leqslant 1) \\ u(0) = 0,\ u(1) = 0 \end{cases}$$

取二项式的近似函数

$$\tilde{u} = \alpha_1 x(1-x) + \alpha_2 x^2(1-x)$$

为近似解. 显然该近似解满足边界条件，但不满足微分方程，产生的余量为

$$R(x) = x + \alpha_1(-2 + x - x^2) + \alpha_2(2 - 6x + x^2 - x^3)$$

其加权积分应为零，即

$$\int_0^1 W_i R \mathrm{d}x = 0$$

(1) 配点法。

取 $x_1 = 1/3$，$x_2 = 2/3$ 作为配点，即有

$$W_1 = \delta\left(x - \frac{1}{3}\right),\quad W_2 = \delta\left(x - \frac{2}{3}\right)$$

代入 $\int_0^1 W_i R \mathrm{d}x = 0$，可得

$$R\left(\frac{1}{3}\right) = \frac{1}{3} - \frac{16}{9}\alpha_1 + \frac{2}{27}\alpha_2 = 0$$

$$R\left(\frac{2}{3}\right) = \frac{2}{3} - \frac{16}{9}\alpha_1 - \frac{50}{27}\alpha_2 = 0$$

求解上述的代数方程可得 $\alpha_1 = 0.1947$，$\alpha_2 = 0.1731$. 于是近似解为

$$\tilde{u} = 0.1947x(1-x) + 0.1731x^2(1-x)$$

(2) 最小二乘法。

取权函数为

$$W_1 = \frac{\partial R}{\partial \alpha_1} = -2 + x - x^2,\quad W_2 = \frac{\partial R}{\partial \alpha_2} = 2 - 6x + x^2 - x^3$$

代入 $\int_0^1 W_j R \mathrm{d}x = \int_0^1 R \dfrac{\partial R}{\partial \alpha_j} \mathrm{d}x = 0$，可得

$$\int_0^1 [x + \alpha_1(-2 + x - x^2) + \alpha_2(2 - 6x + x^2 - x^3)](-2 + x - x^2)\mathrm{d}x = 0$$

$$\int_0^1 [x + \alpha_1(-2 + x - x^2) + \alpha_2(2 - 6x + x^2 - x^3)](2 - 6x + x^2 - x^3)\mathrm{d}x = 0$$

求解上述的代数方程可得 $\alpha_1 = 0.1875$，$\alpha_2 = 0.1695$. 于是近似解为

$$\tilde{u} = 0.1875x(1-x) + 0.1695x^2(1-x)$$

(3) Galerkin 法。

取近似函数序列作为权函数. 由于

$$\tilde{u} = N_1\alpha_1 + N_2\alpha_2 = \alpha_1 x(1-x) + \alpha_2 x^2(1-x) = N\alpha$$

其中，$N = [N_1 \quad N_2]$，$\boldsymbol{\alpha} = [\alpha_1 \quad \alpha_2]^{\mathrm{T}}$.

因而有

$$W_1 = N_1 = x(1-x), \quad W_2 = N_2 = x^2(1-x)$$

代入 $\int_0^1 W_i R \mathrm{d}x = 0$，可得

$$\int_0^1 x(1-x)[x+\alpha_1(-2+x-x^2)+\alpha_2(2-6x+x^2-x^3)]\mathrm{d}x = 0$$

$$\int_0^1 x^2(1-x)[x+\alpha_1(-2+x-x^2)+\alpha_2(2-6x+x^2-x^3)]\mathrm{d}x = 0$$

求解上述的代数方程可得 $\alpha_1 = 0.1924$，$\alpha_2 = 0.1707$. 于是近似解为

$$\tilde{u} = 0.1924x(1-x) + 0.1707x^2(1-x)$$

对于该问题，通过分部积分，其近似等效积分的“弱”形式为

$$J = \int_0^1 W_j\left(\frac{\mathrm{d}^2\tilde{u}}{\mathrm{d}x^2} + \tilde{u} + x\right)\mathrm{d}x = \int_0^1\left(-\frac{\mathrm{d}W_j}{\mathrm{d}x}\frac{\mathrm{d}\tilde{u}}{\mathrm{d}x} + W_j\tilde{u} + W_j x\right)\mathrm{d}x + \left[W_j\frac{\mathrm{d}\tilde{u}}{\mathrm{d}x}\right]_0^1 = 0 \tag{1}$$

由于近似函数为

$$\tilde{u} = N_1\alpha_1 + N_2\alpha_2 = \boldsymbol{N\alpha}, \quad W_1 = N_1 = x(1-x), \quad W_2 = N_2 = x^2(1-x)$$

式(1)可写为

$$\boldsymbol{K\alpha} = \boldsymbol{P}$$

其中

$$\boldsymbol{K} = \int_0^1\left[\left(\frac{\mathrm{d}\boldsymbol{N}}{\mathrm{d}x}\right)^{\mathrm{T}}\frac{\mathrm{d}\boldsymbol{N}}{\mathrm{d}x} - \boldsymbol{N}^{\mathrm{T}}\boldsymbol{N}\right]\mathrm{d}x, \quad \boldsymbol{P} = \int_0^1 \boldsymbol{N}^{\mathrm{T}} x \mathrm{d}x$$

1.2.2 变分方法

对于一个连续介质问题，未知函数 u 的泛函为

$$\Pi = \int_\Omega \boldsymbol{F}\left(u, \frac{\partial u}{\partial x}, \cdots\right)\mathrm{d}\Omega + \int_\Gamma \boldsymbol{E}\left(u, \frac{\partial u}{\partial x}, \cdots\right)\mathrm{d}\Gamma \tag{1-16}$$

其中，$\boldsymbol{F}$ 和 $\boldsymbol{E}$ 是特定的算子，Ω 是求解域，Γ 是 Ω 的边界.

在变分方法中，连续介质问题的解 u 是使泛函 Π 对于微小变化的 δu 取驻值，即泛函的“变分”等于零

$$\delta\Pi = 0 \tag{1-17}$$

对于可以运用变分原理的问题，可以建立其得到近似解的如下方法. 未知函数的近似解可表示成带有待定参数的试探函数

$$u \approx \tilde{u} = \sum_{i=1}^{n} N_i\alpha_i = \boldsymbol{N\alpha} \tag{1-18}$$

式中，α_i 为待定参数，N_i 是已知的函数序列. 将式(1-18)代入式(1-16)，得到用试探函数 $\tilde{u}$ 和待定参数 $\boldsymbol{\alpha}$ 表示的泛函 Π. 泛函的变分为零相当于将泛函对关于待定参数进行全微分，并令其等于零，即

$$\delta\Pi = \frac{\partial\Pi}{\partial\alpha_1}\delta\alpha_1 + \frac{\partial\Pi}{\partial\alpha_2}\delta\alpha_2 + \cdots + \frac{\partial\Pi}{\partial\alpha_n}\delta\alpha_n = 0 \tag{1-19}$$

由于$\delta\alpha_1,\delta\alpha_2,\cdots,\delta\alpha_n$是任意的，式(1-19)成立时必有$\frac{\partial\Pi}{\partial\alpha_1},\frac{\partial\Pi}{\partial\alpha_2},\cdots,\frac{\partial\Pi}{\partial\alpha_n}$都等于零，因而有

$$\frac{\partial\Pi}{\partial\boldsymbol{\alpha}}=\begin{Bmatrix}\frac{\partial\Pi}{\partial\alpha_1}\\ \frac{\partial\Pi}{\partial\alpha_2}\\ \vdots\\ \frac{\partial\Pi}{\partial\alpha_n}\end{Bmatrix}=0 \tag{1-20}$$

由上述与待定参数的数目相等的方程组可求出$\boldsymbol{\alpha}$.

如果在泛函Π中u及其导数的最高阶次为二阶，则称泛函Π为二次泛函. 工程中的许多问题都属于二次泛函. 对于二次泛函问题，式(1-20)退化为一线性方程组

$$\frac{\partial\Pi}{\partial\boldsymbol{\alpha}}=\boldsymbol{K\alpha}-\boldsymbol{P}=0 \tag{1-21}$$

对式(1-21)变分，得

$$\delta\left(\frac{\partial\Pi}{\partial\boldsymbol{\alpha}}\right)=\begin{bmatrix}\frac{\partial}{\partial\alpha_1}\left(\frac{\partial\Pi}{\partial\alpha_1}\right)\delta\alpha_1+\frac{\partial}{\partial\alpha_2}\left(\frac{\partial\Pi}{\partial\alpha_1}\right)\delta\alpha_2+\cdots\\ \vdots\\ \frac{\partial}{\partial\alpha_1}\left(\frac{\partial\Pi}{\partial\alpha_n}\right)\delta\alpha_1+\frac{\partial}{\partial\alpha_2}\left(\frac{\partial\Pi}{\partial\alpha_n}\right)\delta\alpha_2+\cdots\end{bmatrix}=\boldsymbol{K}\delta\boldsymbol{\alpha} \tag{1-22}$$

由矩阵$\boldsymbol{K}$的子矩阵

$$\boldsymbol{K}_{ij}=\frac{\partial^2\Pi}{\partial\alpha_i\partial\alpha_j},\quad \boldsymbol{K}_{ji}=\frac{\partial^2\Pi}{\partial\alpha_j\partial\alpha_i} \tag{1-23}$$

可知

$$\boldsymbol{K}_{ij}=\boldsymbol{K}_{ji}^{\mathrm{T}} \tag{1-24}$$

即矩阵$\boldsymbol{K}$是对称矩阵.

由式(1-21)可将近似泛函表示成

$$\Pi=\frac{1}{2}\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{K\alpha}-\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{P} \tag{1-25}$$

这是因为对式(1-25)中的Π变分，得

$$\delta\Pi=\frac{1}{2}\delta\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{K\alpha}+\frac{1}{2}\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{K}\delta\boldsymbol{\alpha}-\delta\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{P}$$

由矩阵$\boldsymbol{K}$的对称性，有

$$\delta\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{K\alpha}=\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{K}\delta\boldsymbol{\alpha}$$

因而

$$\delta\Pi=\delta\boldsymbol{\alpha}^{\mathrm{T}}(\boldsymbol{K\alpha}-\boldsymbol{P})=0$$

因为$\delta\boldsymbol{\alpha}$是任意的，可得$\boldsymbol{K\alpha}-\boldsymbol{P}=0$，即为式(1-25).

【例 1.2】 对于问题

$$\begin{cases}\dfrac{\partial^2 u}{\partial x^2}+u=-x & (0\leqslant x\leqslant 1)\\ u(0)=0,\ \ u(1)=0\end{cases}$$

建立其变分原理.

该问题的变分可表示为

$$\delta\Pi=\int_0^1\left(\frac{d^2u}{dx^2}-u\quad x\right)\delta u dx+\left[\frac{du}{dx}\delta u\right]_0^1$$

对方程中积分的第一项进行分部积分，可得

$$\delta\Pi=\int_0^1\left(\frac{du}{dx}\frac{d(\delta u)}{dx}-u\delta u-x\delta u\right)dx$$

利用变分算子的交换性

$$\delta\Pi=\delta\int_0^1\left(\frac{1}{2}\left(\frac{du}{dx}\right)^2-\frac{1}{2}u^2-xu\right)dx$$

于是可得到泛函

$$\Pi=\int_0^1\left(\frac{1}{2}\left(\frac{du}{dx}\right)^2-\frac{1}{2}u^2-xu\right)dx$$

1.2.3 Ritz 法

Ritz 法是从一族假定解中寻找满足泛函变分的最好解的近似方法. 在该方法中，近似解的精度与试探函数的选择有关. 如果对所求解的性质了解，则可通过选择反映解特性的试探函数来提高近似解的精度. 若精确解能够包含在试探函数中，则 Ritz 法可求得精确解.

Ritz 法求解的过程与变分方法相同. 这里仍以例 1.1 中的算例来讨论.

【例 1.3】 对于问题

$$\begin{cases}\dfrac{\partial^2 u}{\partial x^2}+u=-x & (0\leqslant x\leqslant 1)\\ u(0)=0,\ \ u(1)=0\end{cases}$$

利用 Ritz 法求近似解.

方程中，边界条件为强制边界条件. 可以选取不同的试探函数形式，用 Ritz 法求解.

(1) 选取近似解为多项式。

$$\tilde{u}=\alpha_1 x(1-x)$$

满足边界条件，且有

$$\frac{d\tilde{u}}{dx}=\alpha_1-2\alpha_1 x$$

将其代入该问题的泛函

$$\Pi-\int_0^1\left(\frac{1}{2}\left(\frac{du}{dx}\right)^2-\frac{1}{2}u^2\quad xu\right)dx$$

可得到用待定参数 α_1 表示的泛函

$$\Pi = \int_0^1 \left[\frac{1}{2}\alpha_1^2 (1-2x)^2 - \frac{1}{2}\alpha_1^2 x^2 (1-x)^2 - \alpha_1 x^2 (1-x) \right] \mathrm{d}x = \frac{1}{2} \times \frac{3}{10}\alpha_1^2 - \frac{1}{12}\alpha_1$$

令泛函变分等于零

$$\frac{\partial \Pi}{\partial \alpha_1} = 0$$

得

$$\alpha_1 = \frac{5}{18}$$

因而近似解为

$$\tilde{u} = \frac{5}{18} x(1-x)$$

(2) 选取近似解为正弦函数式。

$$\tilde{u} = \alpha_1 (\sin x - x \sin 1)$$

满足边界条件，且有

$$\frac{\mathrm{d}\tilde{u}}{\mathrm{d}x} = \alpha_1 (\cos x - \sin 1)$$

将其代入该问题的泛函

$$\Pi = \int_0^1 \left(\frac{1}{2} \left(\frac{\mathrm{d}u}{\mathrm{d}x} \right)^2 - \frac{1}{2} u^2 - xu \right) \mathrm{d}x$$

可得到用待定参数 α_1 表示的泛函

$$\begin{aligned} \Pi &= \int_0^1 \left[\frac{1}{2}\alpha_1^2 (\cos x - \sin 1)^2 - \frac{1}{2}\alpha_1^2 (\sin x - x\sin 1)^2 - \alpha_1 x (\sin x - x \sin 1) \right] \mathrm{d}x \\ &= \frac{1}{2}\alpha_1^2 \sin 1 \left(\frac{2}{3}\sin 1 - \cos 1 \right) - \alpha_1 \left(\frac{2}{3}\sin 1 - \cos 1 \right) \end{aligned}$$

令泛函变分等于零

$$\frac{\partial \Pi}{\partial \alpha_1} = 0$$

得

$$\alpha_1 = \frac{1}{\sin 1}$$

因而近似解为

$$\tilde{u} = \frac{\sin x}{\sin 1} - x$$

上式等于精确解，其原因是所选取的试探函数正好包含了该问题的精确解.

1.2.4　能量原理

在线弹性结构计算中，通常要用到基于变分原理的最小势能能量原理和最小余能原理，在此给出有关的表达式.

1. 弹性力学的基本方程

在线弹性力学中，在载荷的作用下弹性体内任一点的应力状态可由6个应力分量σ_x、σ_y、σ_z、τ_{xy}、τ_{yz}和τ_{zx}来表示，写成矩阵形式为

$$\boldsymbol{\sigma}=\begin{Bmatrix}\sigma_x\\ \sigma_y\\ \sigma_z\\ \tau_{xy}\\ \tau_{yz}\\ \tau_{zx}\end{Bmatrix}=\begin{bmatrix}\sigma_x & \sigma_y & \sigma_z & \tau_{xy} & \tau_{yz} & \tau_{zx}\end{bmatrix}^{\mathrm{T}} \tag{1-26}$$

弹性体内任一点的位移可由沿直角坐标轴方向的3个位移分量u、v和w来表示，矩阵形式为

$$\boldsymbol{u}=\begin{Bmatrix}u\\ v\\ w\end{Bmatrix}=\begin{bmatrix}u & v & w\end{bmatrix}^{\mathrm{T}} \tag{1-27}$$

弹性体内任一点的应变可由6个应变分量ε_x、ε_y、ε_z、γ_{xy}、γ_{yz}和γ_{zx}来表示，矩阵形式为

$$\boldsymbol{\varepsilon}=\begin{Bmatrix}\varepsilon_x\\ \varepsilon_y\\ \varepsilon_z\\ \gamma_{xy}\\ \gamma_{yz}\\ \gamma_{zx}\end{Bmatrix}=\begin{bmatrix}\varepsilon_x & \varepsilon_y & \varepsilon_z & \gamma_{xy} & \gamma_{yz} & \gamma_{zx}\end{bmatrix}^{\mathrm{T}} \tag{1-28}$$

在弹性体V域内的平衡方程为

$$\begin{aligned}&\frac{\partial\sigma_x}{\partial x}+\frac{\partial\tau_{xy}}{\partial y}+\frac{\partial\tau_{xz}}{\partial z}+X=0\\ &\frac{\partial\tau_{yx}}{\partial x}+\frac{\partial\sigma_y}{\partial y}+\frac{\partial\tau_{yz}}{\partial z}+Y=0\\ &\frac{\partial\tau_{zx}}{\partial x}+\frac{\partial\tau_{zy}}{\partial y}+\frac{\partial\sigma_z}{\partial z}+Z=0\end{aligned} \tag{1-29}$$

且有

$$\tau_{xy}=\tau_{yx}，\quad \tau_{yz}=\tau_{zy}，\quad \tau_{zx}=\tau_{xz} \tag{1-30}$$

式中，X、Y和Z为单位体积的体积力在坐标轴上的分量.

力学边界条件(在力边界S_σ上)为

$$\begin{aligned}&\sigma_x l+\tau_{xy}m+\tau_{xz}n-\bar{X}=0\\ &\tau_{yx}l+\sigma_y m+\tau_{yz}n-\bar{Y}=0\\ &\tau_{zx}l+\tau_{zy}m+\sigma_z n-\bar{Z}=0\end{aligned} \tag{1-31}$$

式中：l、m 和 n 为弹性体边界外法线的方向余弦；$\overline{X}$、$\overline{Y}$ 和 $\overline{Z}$ 为作用在单位面积上的面积力在坐标轴上的分量.

几何方程为

$$\varepsilon_x-\frac{\partial u}{\partial x}=0,\quad \varepsilon_y-\frac{\partial v}{\partial y}=0,\quad \varepsilon_z-\frac{\partial w}{\partial z}=0,$$

$$\gamma_{xy}-\left(\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}\right)=0,\quad \gamma_{yz}-\left(\frac{\partial w}{\partial y}+\frac{\partial v}{\partial z}\right)=0,\quad \gamma_{zx}-\left(\frac{\partial u}{\partial z}+\frac{\partial w}{\partial x}\right)=0 \tag{1-32}$$

位移边界条件(在位移边界 S_u 上)为

$$u=\overline{u},\quad v=\overline{v},\quad w=\overline{w} \tag{1-33}$$

本构关系为

$$\begin{Bmatrix}\sigma_x\\ \sigma_y\\ \sigma_z\\ \tau_{xy}\\ \tau_{yz}\\ \tau_{zx}\end{Bmatrix}-\begin{bmatrix}d_{11} & d_{21} & \cdots & d_{61}\\ d_{12} & d_{22} & \cdots & d_{62}\\ \vdots & \vdots & & \vdots\\ d_{16} & d_{26} & \cdots & d_{66}\end{bmatrix}\begin{Bmatrix}\varepsilon_x\\ \varepsilon_y\\ \varepsilon_z\\ \gamma_{xy}\\ \gamma_{yz}\\ \gamma_{zx}\end{Bmatrix}=0 \tag{1-34}$$

对于各向同性体

$$\boldsymbol{D}=\begin{bmatrix}d_{11} & d_{21} & \cdots & d_{61}\\ d_{12} & d_{22} & \cdots & d_{62}\\ \vdots & \vdots & & \vdots\\ d_{16} & d_{26} & \cdots & d_{66}\end{bmatrix}$$

$$=\frac{E(1-\nu)}{(1+\nu)(1-2\nu)}\begin{bmatrix}1 & \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 0 & 0 & 0\\ \frac{\nu}{1-\nu} & 1 & \frac{\nu}{1-\nu} & 0 & 0 & 0\\ \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} & 0 & 0\\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} & 0\\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)}\end{bmatrix} \tag{1-35}$$

式中，E 为弹性模量，ν 为 Poisson(泊松)比.

2. 虚功原理和最小势能原理

根据广义力和广义位移之间的对应关系，将式(1-29)和(1-31)乘上相应的虚位移——任意的、微小的、约束所允许的位移，然后积分并相加，有

$$\iiint_V [(\frac{\partial \sigma_x}{\partial x}+\frac{\partial \tau_{xy}}{\partial y}+\frac{\partial \tau_{xz}}{\partial z}+X)\delta u+(\frac{\partial \tau_{yx}}{\partial x}+\frac{\partial \sigma_y}{\partial y}+\frac{\partial \tau_{yz}}{\partial z}+Y)\delta v$$
$$+(\frac{\partial \tau_{zx}}{\partial x}+\frac{\partial \tau_{zy}}{\partial y}+\frac{\partial \sigma_z}{\partial z}+Z)\delta w]\mathrm{d}V+\iint_{S_\sigma}[(\sigma_x l+\tau_{xy}m+\tau_{xz}n-\bar{X})\delta u$$
$$+(\tau_{yx}l+\sigma_y m+\tau_{yz}n-\bar{Y})\delta v+(\tau_{zx}l+\tau_{zy}m+\sigma_z n-\bar{Z})\delta w]\mathrm{d}S=0 \tag{1-36}$$

应用 Green 定理

$$-\iiint_V \frac{\partial \sigma_x}{\partial x}\delta u\mathrm{d}V=-\iint_{S_\sigma+S_u}\sigma_x l\delta u\mathrm{d}S+\iiint_V \sigma_x\frac{\partial \delta u}{\partial x}\mathrm{d}V$$
$$-\iiint_V \frac{\partial \tau_{xy}}{\partial y}\delta u\mathrm{d}V=-\iint_{S_\sigma+S_u}\tau_{xy}m\delta u\mathrm{d}S+\iiint_V \tau_{xy}\frac{\partial \delta u}{\partial y}\mathrm{d}V$$
$$\vdots$$
$$-\iiint_V \frac{\partial \sigma_z}{\partial z}\delta w\mathrm{d}V=-\iint_{S_\sigma+S_u}\sigma_z n\delta w\mathrm{d}S+\iiint_V \sigma_z\frac{\partial \delta w}{\partial z}\mathrm{d}V$$

可得

$$\iiint_V [(\sigma_x\frac{\partial \delta u}{\partial x}+\tau_{xy}\frac{\partial \delta u}{\partial y}+\tau_{xz}\frac{\partial \delta u}{\partial z}-X\delta u)+(\tau_{yx}\frac{\partial \delta v}{\partial x}+\sigma_y\frac{\partial \delta v}{\partial y}+\tau_{yz}\frac{\partial \delta v}{\partial z}-Y\delta v)$$
$$+(\tau_{zx}\frac{\partial \delta w}{\partial x}+\tau_{zy}\frac{\partial \delta w}{\partial y}+\sigma_z\frac{\partial \delta w}{\partial z}-Z\delta w)]\mathrm{d}V-\iint_{S_u}[(\sigma_x l+\tau_{xy}m+\tau_{xz}n)\delta u$$
$$+(\tau_{yx}l+\sigma_y m+\tau_{yz}n)\delta v+(\tau_{zx}l+\tau_{zy}m+\sigma_z n)\delta w]\mathrm{d}S-\iint_{S_\sigma}(\bar{X}\delta u+\bar{Y}\delta v+\bar{Z}\delta w)\mathrm{d}S=0 \tag{1-37}$$

将式(1-32)和式(1-33)代入式(1-37)，得到

$$\iiint_V [(\sigma_x\delta\varepsilon_x+\sigma_y\delta\varepsilon_y+\sigma_z\delta\varepsilon_z+\tau_{xy}\delta\gamma_{xy}+\tau_{yz}\delta\gamma_{yz}+\tau_{zx}\delta\gamma_{zx})$$
$$-(X\delta u+Y\delta v+Z\delta w)]\mathrm{d}V-\iint_{S_\sigma}(\bar{X}\delta u+\bar{Y}\delta v+\bar{Z}\delta w)\mathrm{d}S=0 \tag{1-38}$$

式(1-38)就是虚功原理的表达式，它表明：当弹性体在外力作用下处于平衡状态时，对于任意为约束所允许的虚位移，外力虚功等于内力虚功.

式(1-38)可写成矩阵形式

$$\iiint_V [\boldsymbol{\sigma}^{\mathrm{T}}\delta\boldsymbol{\varepsilon}-\boldsymbol{F}^{\mathrm{T}}\delta\boldsymbol{u}]\mathrm{d}V-\iint_{S_\sigma}\boldsymbol{P}^{\mathrm{T}}\delta\boldsymbol{u}\mathrm{d}S=0 \tag{1-39}$$

式中：$\boldsymbol{\sigma}$ 为应力向量；$\boldsymbol{\varepsilon}$ 为应变向量；$\boldsymbol{F}$ 为体力向量；$\boldsymbol{P}$ 为面力向量；$\boldsymbol{u}$ 为位移向量.

将本构关系式(1-34)代入式(1-39)，可得

$$\iiint_V [\boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{D}\delta\boldsymbol{\varepsilon}-\boldsymbol{F}^{\mathrm{T}}\delta\boldsymbol{u}]\mathrm{d}V-\iint_{S_\sigma}\boldsymbol{P}^{\mathrm{T}}\delta\boldsymbol{u}\mathrm{d}S=0 \tag{1-40}$$

进而可得

$$\delta\iiint_V [\frac{1}{2}\boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{\varepsilon}-\boldsymbol{F}^{\mathrm{T}}\boldsymbol{u}]\mathrm{d}V-\delta\iint_{S_\sigma}\boldsymbol{P}^{\mathrm{T}}\boldsymbol{u}\mathrm{d}S=0 \tag{1-41}$$

式(1-41)可表示为一个泛函的驻值问题

$$\Pi=\iiint_V [\frac{1}{2}\boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{\varepsilon}-\boldsymbol{F}^{\mathrm{T}}\boldsymbol{u}]\mathrm{d}V-\iint_{S_\sigma}\boldsymbol{P}^{\mathrm{T}}\boldsymbol{u}\mathrm{d}S \tag{1-42}$$

这就是最小势能原理.

通常将

$$U = \iiint_V \frac{1}{2}\boldsymbol{\varepsilon}^{\mathrm{T}} \boldsymbol{D}\boldsymbol{\varepsilon} \mathrm{d}V \tag{1-43}$$

称为内力势能或应变能，将

$$V = -\iiint_V \boldsymbol{F}^{\mathrm{T}} \boldsymbol{u} \mathrm{d}V - \iint_{S_\sigma} \boldsymbol{P}^{\mathrm{T}} \boldsymbol{u} \mathrm{d}S \tag{1-44}$$

称为外力势能，而 $\Pi = U + V$ 称为总势能.

3. 余虚功原理和最小余能原理

根据广义力和广义位移之间的对应关系，将式(1-32)、式(1-33)乘上相应的虚应力——任意的、微小的、平衡所允许的应力，然后积分并相加，有

$$\begin{aligned}
&\iiint_V [(\varepsilon_x - \frac{\partial u}{\partial x})\delta\sigma_x + (\varepsilon_y - \frac{\partial v}{\partial y})\delta\sigma_y + (\varepsilon_z - \frac{\partial w}{\partial z})\delta\sigma_z \\
&+(\gamma_{xy} - \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y})\delta\tau_{xy} + (\gamma_{yz} - \frac{\partial w}{\partial y} - \frac{\partial v}{\partial z})\delta\tau_{yz} + (\gamma_{zx} - \frac{\partial w}{\partial x} - \frac{\partial u}{\partial z})\gamma_{zx}]\mathrm{d}V \\
&+\iint_{S_u} [(u - \bar{u})(\delta\sigma_x l + \delta\tau_{xy} m + \delta\tau_{zx} n) + (v - \bar{v})(\delta\tau_{yz} l + \delta\sigma_y m + \delta\tau_{yz} n) \\
&+(w - \bar{w})(\delta\tau_{zx} l + \delta\tau_{xy} m + \delta\sigma_z n)]\mathrm{d}S = 0
\end{aligned} \tag{1-45}$$

应用 Green(格林)定理

$$-\iiint_V \frac{\partial u}{\partial x}\delta\sigma_x \mathrm{d}V = -\iint_{S_\sigma + S_u} u\delta\sigma_x l \mathrm{d}S + \iiint_V u\frac{\partial \delta\sigma_x}{\partial x}\mathrm{d}V$$

$$-\iiint_V \frac{\partial v}{\partial y}\delta\sigma_y \mathrm{d}V = -\iint_{S_\sigma + S_u} v\delta\sigma_y m \mathrm{d}S + \iiint_V v\frac{\partial \delta\sigma_y}{\partial y}\mathrm{d}V$$

$$-\iiint_V \frac{\partial w}{\partial z}\delta\sigma_z \mathrm{d}V = -\iint_{S_\sigma + S_u} w\delta\sigma_z n \mathrm{d}S + \iiint_V w\frac{\partial \delta\sigma_z}{\partial z}\mathrm{d}V$$

$$\vdots$$

$$-\iiint_V \frac{\partial w}{\partial x}\delta\tau_{zx} \mathrm{d}V = -\iint_{S_\sigma + S_u} w\delta\tau_{zx} l \mathrm{d}S + \iiint_V w\frac{\partial \delta\tau_{zx}}{\partial x}\mathrm{d}V$$

可得

$$\begin{aligned}
&\iiint_V [\varepsilon_x\delta\sigma_x + \varepsilon_y\delta\sigma_y + \varepsilon_z\delta\sigma_z + \gamma_{xy}\delta\tau_{xy} + \gamma_{yz}\delta\tau_{yz} + \gamma_{zx}\delta\tau_{zx} \\
&+u(\frac{\partial\delta\sigma_x}{\partial x} + \frac{\partial\delta\tau_{xy}}{\partial y} + \frac{\partial\delta\tau_{zx}}{\partial z}) + v(\frac{\partial\delta\tau_{yz}}{\partial x} + \frac{\partial\delta\sigma_y}{\partial y} + \frac{\partial\delta\tau_{yz}}{\partial z}) \\
&+W(\frac{\partial\delta\tau_{zx}}{\partial x} + \frac{\partial\delta\tau_{xy}}{\partial y} + \frac{\partial\delta\sigma_z}{\partial z})]\mathrm{d}V - \iint_{S_\sigma} [u(\delta\sigma_x l + \delta\tau_{xy} m + \delta\tau_{zx} n) \\
&+v(\delta\tau_{yz} l + \delta\sigma_y m + \delta\tau_{yz} n) + w(\delta\tau_{zx} l + \delta\tau_{xy} m + \delta\sigma_z n)]\mathrm{d}S \\
&-\iint_{S_u} \bar{u}(\delta\sigma_x l + \delta\tau_{xy} m + \delta\tau_{zx} n) + \bar{v}(\delta\tau_{yz} l + \delta\sigma_y m + \delta\tau_{yz} n) \\
&+\bar{w}(\delta\tau_{zx} l + \delta\tau_{xy} m + \delta\sigma_z n)]\mathrm{d}S = 0
\end{aligned} \tag{1-46}$$

对平衡方程(1-29)和力学边界条件(1-31)进行变分，有

$$
\begin{aligned}
&\frac{\partial\delta\sigma_x}{\partial x}+\frac{\partial\delta\tau_{xy}}{\partial y}+\frac{\partial\delta\tau_{xz}}{\partial z}=0\\
&\frac{\partial\delta\tau_{yx}}{\partial x}+\frac{\partial\delta\sigma_y}{\partial y}+\frac{\partial\delta\tau_{yz}}{\partial z}=0\\
&\frac{\partial\delta\tau_{zx}}{\partial x}+\frac{\partial\delta\tau_{zy}}{\partial y}+\frac{\partial\delta\sigma_z}{\partial z}=0
\end{aligned}
\tag{1-47}
$$

$$
\begin{aligned}
&\delta\sigma_x l+\delta\tau_{xy}m+\delta\tau_{xz}n=0\\
&\delta\tau_{yx}l+\delta\sigma_y m+\delta\tau_{yz}n=0\\
&\delta\tau_{zx}l+\delta\tau_{zy}m+\delta\sigma_z n=0
\end{aligned}
\tag{1-48}
$$

代入式(1-46)，得

$$
\begin{aligned}
&\iiint_V[\varepsilon_x\delta\sigma_x+\varepsilon_y\delta\sigma_y+\varepsilon_z\delta\sigma_z+\gamma_{xy}\delta\tau_{xy}+\gamma_{yz}\delta\tau_{yz}+\gamma_{zx}\delta\tau_{zx}]\mathrm{d}V\\
&-\iint_{S_u}\bar{u}(\delta\sigma_x l+\delta\tau_{xy}m+\delta\tau_{xz}n)+\bar{v}(\delta\tau_{yx}l+\delta\sigma_y m+\delta\tau_{yz}n)\\
&+\bar{w}(\delta\tau_{zx}l+\delta\tau_{zy}m+\delta\sigma_z n)]\mathrm{d}S=0
\end{aligned}
\tag{1-49}
$$

式(1-49)就是余虚功原理的表达式，它表明：当物体处于变形协调状态时，其内力余虚功等于外力余虚功.

式(1-49)可写成矩阵形式

$$\iiint_V \boldsymbol{\varepsilon}^{\mathrm{T}}\delta\boldsymbol{\sigma}\mathrm{d}V-\iint_{S_u}\boldsymbol{u}^{\mathrm{T}}\delta\boldsymbol{P}\mathrm{d}S=0 \tag{1-50}$$

将本构关系式(1-34)代入式(1-50)，可得

$$\iiint_V \boldsymbol{\sigma}^{\mathrm{T}}\boldsymbol{D}^{-1}\delta\boldsymbol{\sigma}\mathrm{d}V-\iint_{S_\sigma}\boldsymbol{u}^{\mathrm{T}}\delta\boldsymbol{P}\mathrm{d}S=0 \tag{1-51}$$

进而可得

$$\delta\iiint_V \frac{1}{2}\boldsymbol{\sigma}^{\mathrm{T}}\boldsymbol{D}^{-1}\boldsymbol{\sigma}\mathrm{d}V-\delta\iint_{S_\sigma}\boldsymbol{u}^{\mathrm{T}}\boldsymbol{P}\mathrm{d}S=0 \tag{1-52}$$

式(1-52)可表示为一个泛函的驻值问题

$$\varGamma=\iiint_V[\frac{1}{2}\boldsymbol{\sigma}^{\mathrm{T}}\boldsymbol{D}^{-1}\boldsymbol{\sigma}\mathrm{d}V-\iint_{S_\sigma}\boldsymbol{u}^{\mathrm{T}}\boldsymbol{P}\mathrm{d}S \tag{1-53}$$

这就是最小余能原理.

通常将

$$U^*=\iiint_V \frac{1}{2}\boldsymbol{\sigma}^{\mathrm{T}}\boldsymbol{D}^{-1}\boldsymbol{\sigma}\,\mathrm{d}V \tag{1-54}$$

称为内力余能，将

$$V^*=-\iint_{S_u}\boldsymbol{u}^{\mathrm{T}}\boldsymbol{P}\,\mathrm{d}S \tag{1-55}$$

称为外力余能，而 $\varGamma=U^*+V^*$ 称为总余能.

1.3　有限元法基本步骤

三维弹性动力学的基本方程是:

平衡方程　$\sigma_{ij,j}+f_i=\rho u_{i,tt}+\mu u_{i,t}$　(在 V 域内)

几何方程　$\varepsilon_{i,j}=\frac{1}{2}\left(u_{i,j}+u_{j,i}\right)$　(在 V 域内)

物理方程　$\sigma_{ij}=D_{ijkl}\varepsilon_{kl}$　(在 V 域内)

边界条件　$u_i=\bar{u}_i$　(在 S_u 边界上)

$\sigma_{ij}n_j=\bar{T}_i$　(在 S_σ 边界上)

初始条件　$u_i(x,y,z,0)=u_i(x,y,z)$

$u_{i,t}(x,y,z,0)=u_{i,t}(x,y,z)$

其中，σ 是应力，u 是位移，ε 是应变，D 是弹性矩阵，f 是体积力，ρ 是质量密度，μ 是阻尼系数，$\bar{T}$ 是面积力，$u_{i,t}$ 和 $u_{i,tt}$ 分别是 u_i 对 t 的一次和二次导数，即 i 方向的速度和加速度，$\rho u_{i,tt}$ 和 $\mu u_{i,t}$ 分别是惯性力和阻尼力，如果是静力学问题，则在平衡方程中就没有这两项.

平衡方程及力的边界条件可以用等效积分的伽辽金形式表示如下

$$\int_V \delta u_i\left(\sigma_{ij,j}+f_i-\rho u_{i,tt}-\mu u_{i,t}\right)\mathrm{d}V-\int_{S_\sigma}\delta u_i\left(\sigma_{ij}n_j-\bar{T}_i\right)\mathrm{d}S=0 \tag{1-56}$$

对式(1-56)的第一项 $\int_V \delta u_i\sigma_{ij,j}\mathrm{d}V$ 进行分部积分，并代入物理方程，可得到

$$\int_V\left(\delta\varepsilon_{ij}D_{ijkl}\varepsilon_{kl}+\delta u_i\rho u_{i,tt}+\delta u_i\mu u_{i,t}\right)\mathrm{d}V=\int_V \delta u_i f_i\mathrm{d}V+\int_{S_\sigma}\delta u_i\bar{T}_i\mathrm{d}S \tag{1-57}$$

由于通过几何方程，应变也可以用位移来表示，所以如果能找到位移在积分区域内的表达式就可以完成上述积分. 这一般是非常困难甚至是不可能的，即使是寻找近似的表达式也是非常困难的. 但考虑到实际问题中位移表达式的函数不会是无限复杂的，如果首先进行对位移空间离散化，只要局部区域划分合适且各局部区域足够小，则在该局部区域中找到具有足够精度的位移关系表达式就比较容易. 这样可以通过分片计算来完成式(1-57)的积分.

一般是通过由区域各节点位移来确定系数的插值函数来近似该区域内的位移函数. 即

$$\left.\begin{aligned}u(x,y,z,t)&=\sum_{i=1}^{n}N_i(x,y,z,t)\,u_i(t)\\ v(x,y,z,t)&=\sum_{i=1}^{n}N_i(x,y,z,t)\,v_i(t)\\ w(x,y,z,t)&=\sum_{i=1}^{n}N_i(x,y,z,t)\,w_i(t)\end{aligned}\right\} \tag{1-58}$$

其中 n 为区域节点数目，$u_i(t),v_i(t),w_i(t)$ 为节点 i 在 t 时刻的位移. 式(1-58)也可以写成

$$u=Na^e \tag{1-59}$$

其中

$$\boldsymbol{u}=\begin{Bmatrix} u(x,y,z,t) \\ v(x,y,z,t) \\ w(x,y,z,t) \end{Bmatrix}, \boldsymbol{N}=\begin{bmatrix} \boldsymbol{N}_1 & \boldsymbol{N}_2 & \cdots & \boldsymbol{N}_n \end{bmatrix}$$

$$\boldsymbol{N}_i = N_i \boldsymbol{I}_{3\times3} \quad (i=1,2,\cdots,n)$$

$$\boldsymbol{a}^e=\begin{Bmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \vdots \\ \boldsymbol{a}_n \end{Bmatrix}, \boldsymbol{a}_i=\begin{Bmatrix} u_i(t) \\ v_i(t) \\ w_i(t) \end{Bmatrix} \quad (i=1,2,\cdots,n)$$

式中，$\boldsymbol{N}$ 为插值函数矩阵或形函数矩阵.

在得到了单元位移函数后，可以利用物理方程来求得单元应变. 将式(1-59)代入几何方程得

$$\boldsymbol{\varepsilon}=\boldsymbol{L}\boldsymbol{u}=\boldsymbol{L}\boldsymbol{N}\boldsymbol{a}^e=\boldsymbol{B}\boldsymbol{a}^e \tag{1-60}$$

式中，$\boldsymbol{B}$ 为应变矩阵，$\boldsymbol{L}$ 为相应问题的微分算子.

通过位移空间离散化，并注意到节点位移变分 $\delta\boldsymbol{a}$ 的任意性，将式(1-59)和式(1-60)代入式(1-57)，就最终可以得到如下的系统求解方程

$$\boldsymbol{M}\ddot{\boldsymbol{a}}(t)+\boldsymbol{C}\dot{\boldsymbol{a}}(t)+\boldsymbol{K}\boldsymbol{a}(t)=\boldsymbol{P}(t) \tag{1-61}$$

式中，$\ddot{\boldsymbol{a}}(t)$ 和 $\dot{\boldsymbol{a}}(t)$ 是系统的节点加速度向量和节点速度向量；$\boldsymbol{M},\boldsymbol{C},\boldsymbol{K}$ 和 $\boldsymbol{P}(t)$ 是系统的质量矩阵、阻尼矩阵、刚度矩阵和节点载荷向量.

在得到上述系统运动方程之后，可以选择用直接积分法或者模态分解法来求解系统响应.

上述动力学有限元方法可以分为 5 步：

(1) 划分区域(划分单元).

(2) 在单个区域内使用近似的位移表达式完成积分. 即计算单元质量矩阵 $\boldsymbol{M}^e=\int_{V_e}\rho\boldsymbol{N}^{\mathrm{T}}\boldsymbol{N}\mathrm{d}V$ 、单元刚度矩阵 $\boldsymbol{K}^e=\int_{V_e}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{B}\mathrm{d}V$ 、单元阻尼矩阵 $\boldsymbol{C}^e=\int_{V_e}\mu\boldsymbol{N}^{\mathrm{T}}\boldsymbol{N}\mathrm{d}V$ 、单元节点载荷向量 $\boldsymbol{P}^e=\int_{V_e}\boldsymbol{N}^{\mathrm{T}}\boldsymbol{f}\mathrm{d}V+\int_{S_\sigma^e}\boldsymbol{N}^{\mathrm{T}}\bar{\boldsymbol{T}}\mathrm{d}S$.

(3) 将所有区域的积分结果求和. 即组装系统整体刚度矩阵 $\boldsymbol{K}=\sum\boldsymbol{K}^e$ 、整体质量矩阵 $\boldsymbol{M}=\sum\boldsymbol{M}^e$ 、整体阻尼矩阵 $\boldsymbol{C}=\sum\boldsymbol{C}^e$ 和整体节点载荷向量 $\boldsymbol{P}=\sum\boldsymbol{P}^e$.

(4) 引入位移边界条件.

(5) 求解系统方程.

相应的有限元程序也可以分为以下几部分.

1.3.1　前处理部分

在这一部分程序中通过定义相应变量来建立模型、划分单元，并对各变量进行说明. 虽然 MATLAB 中的变量不需要定义就可以使用，但是建议通过变量赋初值的方法在程序前部统一建立各变量并加以说明，这样将有助于提高程序的可读性.

作为有限元程序的前处理部分，如果要求实现互动化和智能化，需要建立模型和划分单元，其中单元划分将会是一个复杂的问题. 例如单元划分需要根据所选择单元类型的特点和结构本身的几何特征综合进行考虑，才能得到好的计算结果. 能适用于各种几何特征结构的通用型高质量单元网格划分程序是很难编写的. 早期的一些动力学有限元专著中对这一部分有较详细的论述，但现在无论是在结构建模还是在单元划分方面，都有了可靠而成熟的商用软件，而且当前使用 MATLAB 编程进行有限元分析的读者一般并不是要进行实际工程问题的分析，而是为了进行理论探索和验证. 此时使用几何特征简单的模型来完成研究工作，不但可以减少不必要的计算工作量，而且也有利于突出问题本质. 这样问题就相对简单，可以略过结构模型建立，使用手动输入各单元节点坐标变量的方法来完成单元划分，其工作任务量也可以接受. 所以建议使用 MATLAB 编程进行有限元分析的读者不要把精力花在追求互动的结构模型建立和自动网格划分算法上. 如果那样，还不如直接使用现成而且可靠的各种商业化有限元软件，如 ANSYS 等.

在程序中需要使用哪些变量来完成对问题的描述，是数据结构问题. 合理的数据结构，即进行合理的变量设计，可以使程序更加简洁. 同一个问题可以有不同的算法，而即使同一问题同一算法，在变量设计上也可以有不同的地方. 所以在这里作为示范给出的例子仅供参考，读者也可以根据自己所研究的问题的特点进行相应的变量设计.

可以通过以下变量来描述结构模型.

- node_number：结构的节点总数.
- element_number：结构的单元总数.
- structural_displacement_number：结构位移总数.
- element_displacement_number：单元位移数.
- node_coordinate：节点坐标数组. node_coordinate(I,1)表示第 I 个节点的 x 坐标，node_coordinate(I,2)表示其 y 坐标，而 node_coordinate(I,3)表示其 z 坐标.
- element_node：单元节点编码数组. element_node(I,n)表示第 I 个单元的第 n 个节点编号.

为求解有限元问题定义以下变量.

- element_stiffness_matrix：单元刚度矩阵数组，存储结构全部单元的单元刚度矩阵. element_stiffness_matrix(I,m,n)表示第 I 个单元的单元刚度矩阵中的第 m 行，第 n 列的元素.
- element_mass_matrix：单元质量矩阵数组. 维数同单元刚度矩阵数组.
- element_damping_matrix：单元阻尼矩阵数组. 维数同单元刚度矩阵数组.
- element_load_vector：单元载荷向量数组. element_load_vector(I,n)表示第 I 个单元第 n 个位移方向上的载荷.
- structural_stiffness_matrix：整体刚度矩阵数组. structural_stiffness_matrix(m,n)表示系统整体刚度矩阵中的第 m 行，第 n 列的元素.
- structural_mass_matrix：系统整体质量矩阵数组.
- structural_damping_matrix：系统整体阻尼矩阵数组.

- structural load vector：系统整体载荷向量数组. structural load_vector(n)表示系统第n个位移方向上的载荷.
- Eigenvalue：特征值数组.
- Eigenvector：特征向量数组.

此外，还需要定义结构材料属性等变量以及一些循环变量，这里不再详述.

1.3.2 计算各单元矩阵和单元节点载荷向量

计算各单元矩阵和所选择的单元类型直接相关，这里将简要介绍其基本特点，在后面章节将会详细介绍对于各种常见单元类型，如何计算其刚度、质量和阻尼矩阵，并计算相应单元的节点载荷向量.

1. 单元刚度矩阵

在1.2节中已经推导得到单元刚度矩阵计算表达式为$\boldsymbol{K}^e=\int_{V_e}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{B}\mathrm{d}V$，其一般计算过程为：

(1) 建立单元坐标系. 虽然可以在结构整体坐标系中完成单元刚度矩阵的计算，但是为了更方便地寻找试探函数和进行积分，一般是根据单元的形状来建立合适的单元坐标系. 这样求出的单元刚度矩阵在应用到后面组装成整体刚度矩阵的过程之前，先要进行坐标变化，成为结构整体坐标系下的单元刚度矩阵. 虽然增加了这个过程，不过一般来看，在计算单元刚度矩阵的过程中采用单元坐标系在总体工作量上还是有优势的.

(2) 选择合适的试探函数. 这个问题归结为寻找合适的位移函数表达式. 一般选用各节点位移的插值函数作为单元内的位移函数表达式，此时需要选择合适的形函数.

(3) 按照$\boldsymbol{K}^e=\int_{V_e}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{B}\mathrm{d}V$完成积分，得到单元刚度矩阵.

现在为了更好地理解单元刚度矩阵的物理意义，可以利用最小位能原理建立一个单元的平衡方程，得到

$$\boldsymbol{K}^e\boldsymbol{a}^e=\boldsymbol{P}^e \tag{1-62}$$

式中，$\boldsymbol{P}^e$是单元节点载荷，当然也包括其他相邻单元对该单元的作用力. $\boldsymbol{P}^e$中的载荷排列顺序与$\boldsymbol{a}^e$中各节点位移的顺序保持一致. 例如

$$\boldsymbol{a}^e=\begin{bmatrix}u_1 & v_1 & w_1 & u_2 & v_2 & w_2 & \cdots\end{bmatrix}^{\mathrm{T}},\quad \boldsymbol{P}^e=\begin{bmatrix}P_{1x} & P_{1y} & P_{1z} & P_{2x} & P_{2y} & P_{2z} & \cdots\end{bmatrix}^{\mathrm{T}} \tag{1-63}$$

令$u_1=1$，$\boldsymbol{a}^e$中其他元素为零，则得到

$$\begin{bmatrix}k_{11}\\ k_{21}\\ \vdots\end{bmatrix}=\begin{Bmatrix}P_{1x}\\ P_{1y}\\ \vdots\end{Bmatrix} \tag{1-64}$$

由此可见，单元刚度矩阵第1列元素的物理意义是：当1号节点的1号位移方向上有单位位移，而其他节点位移全为零时，需要在单元各节点位移方向上施加的节点力的大小. 相应地也可以得到其他列元素的物理意义. 因此单元刚度矩阵中任一元素k_{ij}的物理意义

为：当单元的第 j 个节点位移为单位位移而其他节点位移为零时，需要在单元第 i 个节点上施加的节点力的大小. 单元刚性大，则使节点产生单位位移所需要施加的节点力就大，因此单元刚度矩阵中的每个元素反应了单元刚性的大小，称为刚度系数.

从其物理意义，可以推得单元刚度矩阵的 3 个特性.

- 对称性：根据材料力学中的位移互等定理，可以证明单元刚度矩阵具有对称性，即 $k_{ij} = k_{ji}$.
- 奇异性：单元处于平衡时，节点力相互不是独立的，它们必须满足单元平衡方程，因此它们是线性相关的. 另一方面，即使给定满足平衡条件的单元节点力 $\boldsymbol{P}^e$，也不能确定单元节点位移 $\boldsymbol{a}^e$，因为单元还可以有任意的刚体位移.
- 主元恒正：即 $k_{ii} > 0$，根据其物理意义，k_{ii} 是结构第 i 个节点且有单元位移，而其他节点位移全为零时，在第 i 个节点方向上所需要施加的节点力，此节点力自然不能为零，或者为负值(与位移方向相反).

2. 单元质量矩阵

在 1.2 节中推导得到的单元质量矩阵 $\boldsymbol{M}^e = \int_{V_e} \rho \boldsymbol{N}^{\mathrm{T}} \boldsymbol{N} \mathrm{d}V$ 称为协调质量矩阵或一致质量矩阵，这是因为导出它和导出单元刚度矩阵所依据的原理及采用的位移插值函数是一致的，同时质量分布也是按照实际分布情况考虑的. 除此之外，还可以采用集中(团聚)质量矩阵，这种质量矩阵是简单地假定单元的质量是集中在节点上，而得到对角线型矩阵.

上述两种质量矩阵在实际工程中都可以应用. 一般情况下，两者所得结果也相近. 单元刚度矩阵积分表达式的被积函数是插值函数的导数的平方项，而一致质量矩阵的积分表达式中的被积函数是插值函数的平方项，因此在相同精度要求下，质量矩阵可以用较低阶的插值函数，而集中质量矩阵从本质上看，正是这样一种替换方案. 采用集中质量矩阵的优点是质量矩阵是对角矩阵，可以在后面的求解过程中使计算得到简化. 此外，对于节点参数中包含转动的单元类型，由于可以在集中质量矩阵中略去转动项，如采用显式直接积分方法求解运动方程，还可以使方程的自由度进一步减少. 采取集中质量矩阵的困难是对于高次单元如何将单元的质量分配到各个节点上.

3. 单元阻尼矩阵

基于和协调质量矩阵相同的理由，称 $\boldsymbol{C}^e = \int_{V_e} \mu \boldsymbol{N}^{\mathrm{T}} \boldsymbol{N} \mathrm{d}V$ 为协调阻尼矩阵. 它是假定阻尼力正比于质点运动速度的结果，通常将介质阻尼简化为这种情况，这时单元阻尼矩阵比例于单元质量矩阵.

除此之外，还有比例于应变速度的阻尼，例如由材料内部摩擦引起的结构阻尼通常可以简化为这种情况，这时阻尼力可以表示为 $\mu \boldsymbol{D} \dot{\varepsilon}$，这样得到的单元阻尼矩阵为

$$\boldsymbol{C}^e = \mu \int_{V_e} \boldsymbol{B}^{\mathrm{T}} \boldsymbol{D} \boldsymbol{B} \mathrm{d}V \tag{1-65}$$

此单元阻尼矩阵比例于单元刚度矩阵.

由于系统的固有振型对于整体刚度矩阵 $\boldsymbol{K}$ 和质量矩阵 $\boldsymbol{M}$ 具有正交性，因此固有振型对于比例于整体刚度矩阵和质量矩阵的阻尼矩阵也是具有正交性的，所有这种阻尼矩阵称

为比例阻尼或者振型阻尼. 可以在后面的求解过程中利用振型矩阵将阻尼矩阵对角化，为计算带来很大的方便.

一般情况下，比例系数是依赖于频率的. 因此在实际分析中，要精确地测定阻尼矩阵是相当困难的. 通常将实际结构的阻尼矩阵简化为整体刚度矩阵和质量矩阵的线性组合，即

$$C = \alpha M + \beta K \tag{1-66}$$

其中α, β是不依赖于频率的常数，这种振型阻尼称为Rayleigh阻尼.

4. 单元节点载荷向量

在前面已经给出了计算单元节点载荷向量的一般公式：

$$P^e = \int_{V_e} N^T f dV + \int_{S_\sigma} N^T \bar{T} dS \tag{1-67}$$

如果单元边界力$\bar{T}$是分布边界力，则代入式(1-67)进行计算；如果单元边界力是集中载荷，则无须再将其等效为单元节点载荷. 可以调整单元网格，使得集中载荷作用的位置正好是一个单元节点，然后在单元节点载荷向量这一部分可以先不必考虑该集中载荷，而是根据其所作用的节点编号和与其方向相同的节点位移方向编号，将其直接叠加到系统整体节点载荷向量中去.

1.3.3 系统整体刚度、质量、阻尼矩阵和节点载荷向量

系统整体刚度、质量、阻尼矩阵和节点载荷向量是由相应的单元矩阵组装而成的，其组装过程是按照单元位移编号和系统整体位移编号的关系来进行的. 下面以整体刚度矩阵的组装过程为例来说明这一过程.

在推导得出单元刚度矩阵表达式$K^e = \int_{V_e} B^T DB dV$时，首先在式(1-59)中对单元各节点位移进行了排序编号，因此单元刚度矩阵中的元素排列是和节点位移编号顺序直接相关的. 在将单元刚度矩阵组装成整体刚度矩阵的过程中也要根据这一对应关系来进行. 如果在建立单元刚度矩阵过程中使用了单元坐标系，那么在组装之前，还需要根据单元坐标和系统坐标间的关系，对单元刚度矩阵进行转换，得到在系统坐标系下的单元刚度矩阵，然后才可以进行组装. 这一转换过程可以表示为

$$K^e = T^T \bar{K}^e T \tag{1-68}$$

式中：K^e为系统坐标系下的单元刚度矩阵；$\bar{K}^e$为单元坐标系下的单元刚度矩阵；T为坐标转换矩阵. 如果在建立单元刚度矩阵的过程中没有使用单元坐标系，则不需要这个过程.

若在单元位移中的编号为1的单元的节点位移在系统位移中编号为i，单元位移中的编号为2的单元的节点位移在系统位移中编号为j，……，则单元刚度矩阵中的元素k_{11}^e应该被叠加到整体刚度矩阵中的元素k_{ii}上，而k_{12}^e则应叠加到k_{ij}上，其他元素都可以按照编号对应关系，依次类推，叠加到整体刚度矩阵中去.

在MATLAB中可以通过如下代码实现由系统坐标系下的单元刚度矩阵组装系统整体刚度矩阵的过程：

```
% ssm: structural stiffness matrix
% esm: element stiffness matrix
% e2s: index of transform the element displament number to structural
displament number
for es=1:element_number
% es(element serial) is the loop variable.
    for ers=1:element_displacement_number
        % ers is the row serial of element stiffness matrix.
        for ecs=1:element_displacement_number
        % ecs is the column serial of element stiffness matrix.
            ssm(e2s(es,ers), e2s(es,ecs))=ssm(e2s(es,ers), e2s(es,ecs))
              +esm(es,ers,ecs);
        end
    end
end
```

包括从单元坐标系到系统坐标系的转换过程在内，整体质量矩阵和阻尼矩阵的组装过程与整体刚度矩阵组装过程相同，所以可以集中在一个循环中解决. 整体载荷向量组装的思想与整体刚度矩阵的组装思想相同，只是要注意整体载荷向量是向量，用一维数组存储，所以不需要进行二重循环. 因此可以在上面的 MATLAB 代码中添加少数语句，再同时实现整体质量矩阵、阻尼矩阵和载荷向量的组装.

```
% ssm: structural stiffness matrix
% esm: element stiffness matrix
% e2s: index of transforming the element displament number to structural
displament number
% smm: structural_mass_matrix
% emm: element_mass_matrix
% sdm: structural_damping_matrix
% edm: element_ damping_matrix
% slv: structural_load_vector
% elv: element_load_vector
for es=1:element_number
    for ers=1:element_displacement_number
        for ecs=1:element_displacement_number
            ssm(e2s(es,ers), e2s(es,ecs))=ssm(e2s(es,ers), e2s(es,ecs))
              +esm(es,ers,ecs);
            smm(e2s(es,ers),e2s(es,ecs))=smm(e2s(es,ers),e2s(es,ecs))
              +emm(es,ers,ecs);
            sdm(e2s(es,ers),e2s(es,ecs)) =sdm(e2s(es,ers),e2s(es,ecs))
              +edm(es,ers,ecs);
        end
      slv(e2s(es,ers))=slv (e2s(es,ers))+elv(es,ers);
    end
end
```

在完成组装过程，得到结构整体矩阵之后，通过对其特性的研究，能够帮助减少整个

求解过程的计算量和所需的计算机存储空间. 下面以结构整体刚度矩阵为例进行说明.

结构整体刚度矩阵是由单元刚度矩阵组装而成的，因此与单元刚度矩阵有类似的物理意义，其主要性质介绍如下.

(1) 对称性：源于单元刚度矩阵的对称性.

(2) 带状特性：在连续体离散为有限个单元后，每个节点的相关单元只是围绕在该单元周围为数不多的几个，一个节点通过相关单元与之发生关系的相关节点也只是它周围的少数几个，这样根据整体刚度矩阵元素的物理意义，为了在结构某个节点位移方向上有单位位移而结构其他节点位移方向上位移全为零，只需要在该节点和其周围关联节点上施加力就可以了，其余无关节点上并不需要有节点力，且无关节点一定是绝大多数，这样在每一列元素中都是零元素占多数. 因此虽然结构总单元数和节点数很多，结构整体刚度阶数很高，但刚度系数中非零元素却很少，因此只要节点编号是合理的，也就是说相邻节点间的编号不要相差过远，这些稀疏的非零元素将集中在以主对角线为中心的一条带状区域内，即具有带状分布的特点.

(3) 稀疏：如上所述，在整体刚度矩阵中应是非零元素占多数，所以具有稀疏的特性.

在得到整体刚度矩阵后，可以通过使用 spy 函数来观察其中非零元素的分布，以此来验证上述结论. spy 函数通过将二维矩阵转换为一个平面图形来形象地显示其中的非零元素分布：在图形中非零元素将在其对应的位置上绘制一个点，而零元素所对应的位置则留作空白. 如在 MATLAB 命令窗口中输入：spy(K). 其中 K 是准备进行观察的整体刚度矩阵，将得到如图 1.1 所示的图形.

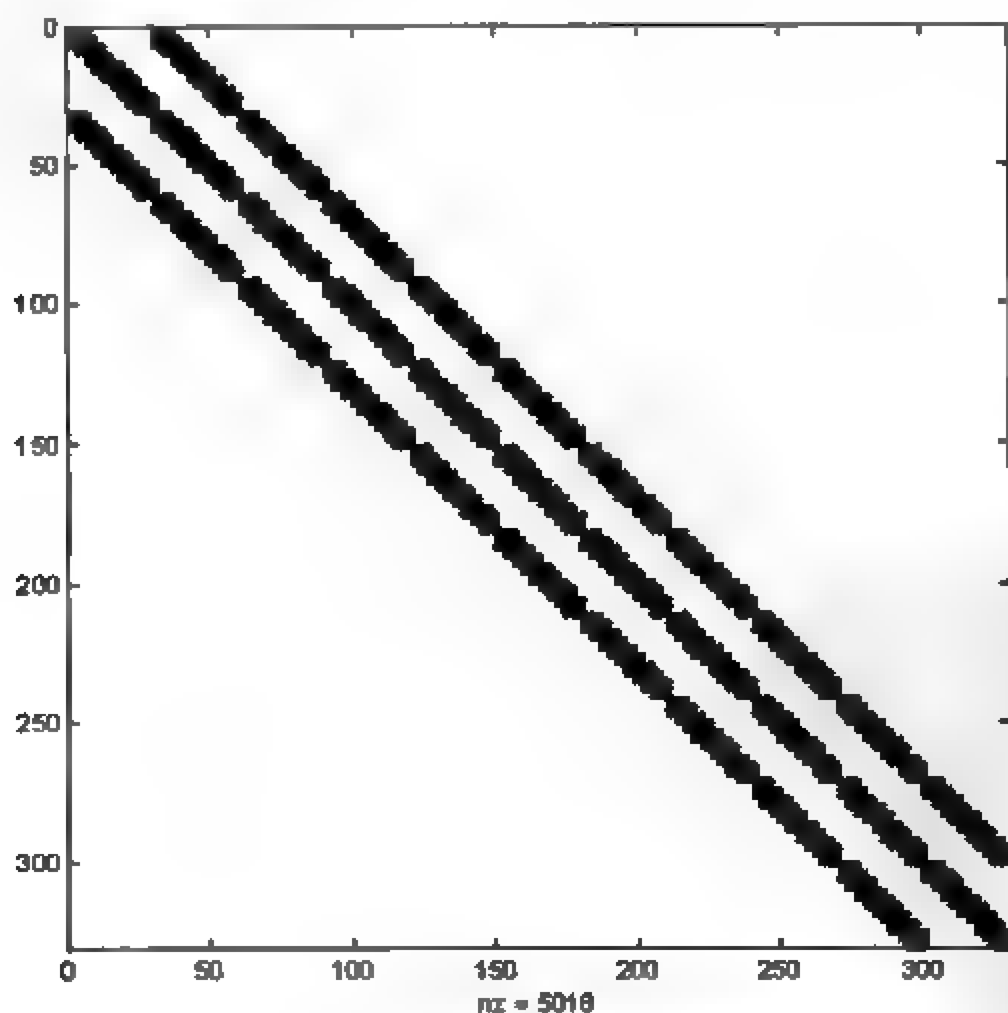


图 1.1　整体刚度矩阵非零元素分布图

可以看出在这个 330 阶的矩阵中，非零元素呈对称带状分布，且在全部 108 900(330×330)个元素中，只有 5 018 个非零元素(图 1.1 中 nz 表示非零元素数目)，仅为元素总数的 4.61%. 随着自由度阶数的增加，这一比例一般还会更低.

整体刚度矩阵的这些特性，可以被用来节约存储整体刚度矩阵所需要的空间. 如利用带状和稀疏特性，可以只存储矩阵中的非零元素. 进一步利用对称性，可以只存储矩阵非零元素中的一半. 这样的存储方法可以极大地减少所需的存储空间，不利之处是增加了编写程序的难度. 但是在 MATLAB 中，却可以方便地使用稀疏矩阵来存储此类矩阵. 只要预先定义了矩阵变量为稀疏矩阵，在 MATLAB 中可以不用去了解稀疏矩阵具体是如何存储的，而是直接按照常规矩阵的使用方法去读取和改变稀疏矩阵的内容. 这样既节约了存储空间，又不带来程序编写的困难.

类似地，整体质量矩阵和整体阻尼矩阵也可以使用稀疏矩阵来存储.

1.3.4 施加位移约束条件

在通过模型离散化得到近似场函数来求解弹性动力学基本方程的过程中，所选择的场函数在单元内部满足几何方程，因此在由离散模型近似的连续体内也满足几何方程. 但是在选择场函数的试探函数时，并没有提出在边界上满足位移边界条件的要求，因此必须将这个条件引入有限元方程，使之得到满足.

在有限元方法中，几何边界条件的形式通常是在若干个节点上给定场函数的值(可以是零值，也可以是非零值)，即

$$a_j = \overline{a}_j \quad (j = c_1, c_2, \cdots, c_l)$$

对于静力学问题，求解位移场问题时，至少要提出足以约束系统刚体位移的几何边界条件，以消除结构整体刚度矩阵的奇异性，一般可以使用以下方法来施加位移约束条件.

1. 直接代入法

在结构动力学方程中将已知节点位移的自由度消去，得到一组修正方程，用以求解其他待定的节点位移，其原理是根据已知和待定的节点位移重新组合方程，可得到

$$\begin{bmatrix} \boldsymbol{K}_{aa} & \boldsymbol{K}_{ab} \\ \boldsymbol{K}_{ba} & \boldsymbol{K}_{bb} \end{bmatrix} \begin{Bmatrix} \boldsymbol{a}_a \\ \boldsymbol{a}_b \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{P}_a \\ \boldsymbol{P}_b \end{Bmatrix} \tag{1-69}$$

式中：$\boldsymbol{a}_a$ 为待定节点位移；$\boldsymbol{a}_b$ 为已知节点位移，$\boldsymbol{a}_b^{\mathrm{T}} = [\overline{a}_{c1} \quad \overline{a}_{c2} \quad \cdots \quad \overline{a}_{cl}]$；$\boldsymbol{K}_{aa}, \boldsymbol{K}_{ab}, \boldsymbol{K}_{ba}, \boldsymbol{K}_{bb}, \boldsymbol{P}_a, \boldsymbol{P}_b$ 为与之相应的刚度矩阵和载荷向量的分块矩阵. 由刚度矩阵的对称性可知 $\boldsymbol{K}_{ba} = \boldsymbol{K}_{ab}^{\mathrm{T}}$.

由式(1-69)的第一式可得

$$\boldsymbol{K}_{aa}\boldsymbol{a}_a + \boldsymbol{K}_{ab}\boldsymbol{a}_b = \boldsymbol{P}_a \tag{1-70}$$

由于 $\boldsymbol{a}_b$ 为已知，最后求解方程可写为

$$\boldsymbol{K}^*\boldsymbol{a}^* = \boldsymbol{P}^* \tag{1-71}$$

其中

$$\boldsymbol{K}^* = \boldsymbol{K}_{aa}, \quad \boldsymbol{a}^* = \boldsymbol{a}_a, \quad \boldsymbol{P}^* = \boldsymbol{P}_a - \boldsymbol{K}_{ab}\boldsymbol{a}_b$$

若总节点位移为 n 个，其中有已知节点位移 m 个，则得到一组求解 $n-m$ 个待定节点位移的修正方程组，K^* 为 $n-m$ 阶方阵. 修正方程组的意义是在原来的 n 个方程中，只保留与待定节点位移相对应的 $n-m$ 个方程，并将方程中左端的已知位移和相应刚度系数的乘积(为已知值)移至方程右端作为载荷修正项.

如果已知节点位移 a_b 全部为零，那么可以直接求解 $K_{aa}a_a = P_a$. 这种重新组合方程的做法可以降低方程阶数，但是在组合过程中，节点位移的顺序被改变，为了正确理解将来的计算结果，必须对结果进行编号“还原”，增加了编程的复杂程度.

2. 对角元素改 1 法

如果已知节点位移 a_b 全部为零，可以在 K 中将与已知零节点位移相对应的行列中的主对角元素改为 1，其他元素改为 0；在载荷向量中也将相应的元素改为 0. 例如，节点位移中的第 j 个元素为已知零位移，那么应将 K 中的第 j 行和第 j 列元素除了 k_{jj} 改为 1 之外，其他全部改为 0；同时将载荷向量中的第 j 个元素也改为 0.

这样对所有已知零位移完成相应的修改，可以在不改变原方程阶数和节点未知量的顺序编号的情况下，简单地引入位移边界条件，但是这种方法只能用于给定零位移.

3. 对角元素乘大数法

如节点位移中的第 j 个元素为已知位移，其值为 $\bar{a}_j$，则可以直接对 k_{jj} 乘以一个很大的数 α (量级可以取 10^{10})，并将载荷向量中的第 j 个元素替换为 $\alpha k_{jj}\bar{a}_j$ 来引入位移边界条件.

这种方法对零值和非零值的给定位移都适用，引入位移边界条件时，也不改变节点位移顺序. 对总自由度较小的结构可以考虑这种方法；而对总自由度较大的结构，可以考虑采取直接代入法来缩小方程阶数，以节省求解时间.

上述为静力学中常用的 3 种引入位移边界条件的方法. 对于动力学问题，求解系统固有频率和振型时，可以不施加任何约束，这样计算得到的固有频率中的前 6 阶应该是接近于零的数值，对应于系统的 6 个刚体位移模态. 在求解响应信号时，一般需要合适的约束以保持系统不至于在该激励下产生刚体位移. 由于动力学问题的计算量一般都远远大于相同自由度数目的静力学问题，因此能够降低计算量的方法具有更加重要的意义. 所以在动力学问题中，一般都采用直接代入法在运动方程中消去约束自由度. 在编程中可以不必先完全组装再消去约束自由度，而是可以在组装之前，就先对系统位移进行重新编号，在新的编号中不包括被约束的位移，然后按照新的编号完成组装过程. 在动力学问题中引入不为零约束的方法比较复杂，可以参考了结构的相关方法进行.

1.3.5 求解

由于动力学有限元问题式(1-61)的巨大求解计算量，相应的求解算法研究一直得到广泛的重视. 根据激励的不同，可以将式(1-61)分为冲击响应、谐波激励响应和瞬态响应(任意激励响应)问题，其中每一种问题都有多种求解算法，而且动力学问题并不仅仅局限于

求解系统响应. 为了更好地理解系统特性，还提出了各种系统动力学特征指标，如频率、模态和频响函数等，在计算这些指标方面也存在着众多的算法. 因此动力学问题的求解部分包含的内容非常广泛. 在经典有限元著作中，往往有相关章节对各种算法进行介绍. 现在使用 MATLAB 时，可以不必对算法进行深入了解而直接通过调用相关函数来完成问题的求解. 这样不但有利于更好地将精力集中于动力学问题本身，而且 MATLAB 中的函数也比读者自己编写的更加高效可靠. 基于这一原因，本书将不再侧重于介绍各种算法的数学本质，而是主要介绍如何使用 MATLAB 实现各种算法.

瞬态响应问题是实际工程和研究中常见的一种问题，且可以将冲击响应和谐波激励响应问题视为瞬态响应问题的特例，所以在这里首先以瞬态响应为例介绍在 MATLAB 中求解动力学问题的一般过程.

对式(1-61)，可以有直接积分法和振型组合法两种方法进行求解. 直接积分法是指在积分运动方程之前不进行方程形式的变换，而直接进行逐步数值积分. 振型组合法是指在积分运动方程之前，利用系统固有振型(模态)的关于刚度矩阵和质量矩阵的正交性对方程组进行解耦，然后再进行解析或者数值积分. 与直接积分法比较，振型组合法的缺点是不能用于非线性问题求解，其优点主要有以下两点.

(1) 积分步长选择上的优点. 在使用直接积分法时，为了保证结果的精度，积分步长必须保证小于系统中最高频率振型的周期. 而完成方程解耦后，在采用数值方法时，对于每个方程可以采取不同的时间步长，即对低阶振型可以采用较大的时间步长.

(2) 可以只求解由少数模态组成的响应. 首先在实际工程问题中，由于结构的高阶模态很难激起，所以一般情况下结构的响应主要是由低阶模态振动部分组成. 其次由于结构的高阶模态振型比低阶模态复杂，而有限元求解过程是对结构进行分区域近似的过程，所以对于高阶模态，有限元方法的精度会相对较低. 综合以上两点，在一般的研究中没有必要求解出响应中的高阶模态部分，而可以根据问题的特点只求解少数几阶的低阶模态，这样在精度允许的条件下，可以大幅地减少计算量.

由于解耦后的方程组中，每一个方程都含有一个变量，方程组实际上是由多个不相关的方程构成. 对其中任意一个都可以单独按照单自由度系统受激振动的方法来求解.

由此可见，对于自由度数目较多，且只关心少数低阶模态响应的问题，振型组合法在减少计算量方面，将具有很大的优势.

第 2 章　结构的动力特性和响应分析

2.1　动力特性分析

2.1.1　矢量迭代法

矢量迭代法是结构动力特性分析中经常使用的方法之一. 矢量迭代法分为逆迭代法和正迭代法两种. 逆迭代法向结构的最低频率和振型收敛，而正迭代法向结构的最高频率和振型收敛. 一般情况下，不使用正迭代法，这一方面是因为在实际工程问题中所感兴趣的是系统的少数低特征对，另一方面是因为以结构的有限单元离散化模型为基础计算结构的高特征对精度很差. 在具体实施矢量迭代法时，可以有各不相同的具体步骤，但它们的基本思想与使用的迭代关系式是相同的.

逆迭代法除了求最小特征值和特征向量外，和 Gram-Schmidt(格拉姆-施密特)正交化过程相结合，可用来求取最低的几阶特征对. 对于第 j 阶特征对的求取，设有初始向量 $\overline{\boldsymbol{x}}_j^{(0)}$ 及其第 $s=1,2,\cdots$ 时迭代公式为

$$\boldsymbol{K}\boldsymbol{x}_j^{(s+1)}=\boldsymbol{M}\boldsymbol{x}_j^{(s)} \tag{2-1}$$

清型(即清理模型)是求取了最小特征对之后求其余特征对时所必须进行的一步，不仅包括特征对求取的迭代初始向量与前 $j-1$ 阶特征向量正交，且每次迭代过程中都要进行正交化处理，不断把前 $j-1$ 阶特征向量从迭代向量中清除掉，因为由于实际计算的误差，在迭代过程中不可避免地会产生低阶特征向量，迭代到最后还是得到最低阶的特征向量. 则 Gram-Schmidt 正交化过程如下

$$\overline{\boldsymbol{x}}_{j+1}^{(0)}=\boldsymbol{x}_{j+1}^{(0)}-\sum_{i=1}^{j}\beta_i\boldsymbol{\phi}_i \tag{2-2}$$

其中

$$\beta_i=\boldsymbol{\phi}_i^{\mathrm{T}}\boldsymbol{M}\boldsymbol{x}_{j+1}^{(0)} \tag{2-3}$$

初始向量选得好与坏，对迭代的效率影响极大. 显然，要求初始向量选得尽可能接近所求的振型. 一种简单的选取方法是，把结构简单地考虑为各个自由度是互不耦合的，即

$$\boldsymbol{K}=\mathrm{diag}\begin{bmatrix}k_{11} & k_{22} & \cdots & k_{nn}\end{bmatrix} \tag{2-4}$$

$$\boldsymbol{M}=\mathrm{diag}\begin{bmatrix}m_{11} & m_{22} & \cdots & m_{nn}\end{bmatrix} \tag{2-5}$$

相应式(2-4)与式(2-5)的特征方程为非耦合形式

$$k_{ii}\delta_i=\lambda_i m_{ii}\delta_i\quad (i=1,2,\cdots,n) \tag{2-6}$$

则有 $\lambda_i=k_{ii}/m_{ii},\delta_i=1$. 将 λ_i 从小到大排序，设 λ_i 排在第 j 位.

一般选取第一阶初始向量为

$$\boldsymbol{M}\boldsymbol{x}_1^{(0)}=\begin{bmatrix}m_{11} & m_{22} & \cdots & m_{nn}\end{bmatrix}^{\mathrm{T}} \tag{2-7}$$

取第 $j+1$ 阶初始向量为

$$\boldsymbol{M}\boldsymbol{x}_{j+1}^{(0)}=\begin{matrix} & 1 & 2 & \cdots & & i & & \cdots & n \\ [& 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 &]^{\mathrm{T}}\end{matrix} \tag{2-8}$$

矢量逆迭代法的计算步骤如下：

选取 p 阶初始 n 维矢量 $\boldsymbol{y}_j^{(0)}(j=1,2,\cdots,p)$

对于 $j=1,2,\cdots,p$ 进行循环

对于 $s=1,2,\cdots$

计算 $\boldsymbol{K}\overline{\boldsymbol{x}}_j^{(s+1)}=\boldsymbol{y}_j^{(s)}$

计算 $\overline{\boldsymbol{y}}_j^{(s+1)}=\boldsymbol{M}\overline{\boldsymbol{x}}_j^{(s+1)}$

计算近似特征值 $\lambda_j^{(s+1)}=\dfrac{\left(\overline{\boldsymbol{x}}_j^{(s+1)}\right)^{\mathrm{T}}\boldsymbol{y}_j^{(s)}}{\left(\overline{\boldsymbol{x}}_j^{(s+1)}\right)^{\mathrm{T}}\overline{\boldsymbol{y}}_j^{(s+1)}}$

如果特征值的精度满足条件 $\left|\dfrac{\lambda_j^{(s+1)}-\lambda_j^{(s)}}{\lambda_j^{(s+1)}}\right|\leqslant\varepsilon$，则终止内循环，计算特征向量 $\boldsymbol{\phi}_j=\dfrac{\overline{\boldsymbol{x}}_j^{(s+1)}}{\left(\left(\overline{\boldsymbol{x}}_j^{(s+1)}\right)^{\mathrm{T}}\overline{\boldsymbol{y}}_j^{(s+1)}\right)^{1/2}}$.

Gram-Schmidt 正交化，$\overline{\boldsymbol{x}}_{j+1}^{(0)}=\boldsymbol{x}_{j+1}^{(0)}-\sum_{i=1}^{j}\beta_i\boldsymbol{\phi}_i$，其中 $\beta_i=\boldsymbol{\phi}_i^{\mathrm{T}}\boldsymbol{M}\boldsymbol{x}_{j+1}^{(0)}$.

否则计算下次迭代的初始向量 $\boldsymbol{y}_j^{(s+1)}=\dfrac{\overline{\boldsymbol{y}}_j^{(s+1)}}{\left(\left(\overline{\boldsymbol{x}}_j^{(s+1)}\right)^{\mathrm{T}}\overline{\boldsymbol{y}}_j^{(s+1)}\right)^{1/2}}$.

2.1.2 子空间迭代法

子空间迭代法的基本思想是把逆迭代法和 Ritz 法结合起来，既利用 Ritz 法来缩减自由度，又在计算过程中利用逆迭代法使振型逐步趋近其精确值，由于它吸收了两种方法的优点，因而计算效果比较好. 经验表明，这是目前求解大型结构自振频率和振型的最有效方法之一.

对于 n 阶实对称矩阵 $\boldsymbol{M}$ 与 $\boldsymbol{K}$，迭代式可取为

$$\boldsymbol{K}\boldsymbol{X}_{k+1}=\boldsymbol{M}\hat{\boldsymbol{X}}_k \tag{2-9}$$

式中：$\boldsymbol{X}_{k+1}$ 是 $n\times m$ 阶矩阵，且 $m<n$；$\hat{\boldsymbol{X}}_k$ 是 $\boldsymbol{X}_k$ 关于 $\boldsymbol{M}$ 的正交归一化矩阵. $\hat{\boldsymbol{X}}_k$ 的正交归一化可进行如下：借变换 $\boldsymbol{X}_k$，经 Ritz 缩聚后，有

$$\boldsymbol{M}_k=\boldsymbol{X}_k^{\mathrm{T}}\boldsymbol{M}\boldsymbol{X}_k,\quad \boldsymbol{K}_k=\boldsymbol{X}_k^{\mathrm{T}}\boldsymbol{K}\boldsymbol{X}_k \tag{2-10}$$

分别记缩聚系统($\boldsymbol{M}_k$，$\boldsymbol{K}_k$)的特征值矩阵与特征矢量矩阵为 $\boldsymbol{\lambda}_k^*$ 与 $\boldsymbol{P}_k$，有

$$\boldsymbol{K}_k\boldsymbol{P}_k=\boldsymbol{M}_k\boldsymbol{P}_k\boldsymbol{\lambda}_k^* \tag{2-11}$$

式中，$\boldsymbol{\lambda}_k^*$ 是对角阵，$\boldsymbol{P}_k$ 是关于 $\boldsymbol{M}_k$ 的正归一化矩阵.

故在下一步迭代中可取

$$\hat{X}_k = X_k P_k \tag{2-12}$$

需要指出的是，在子空间迭代法中，初始向量的选取也将直接影响到迭代的收敛速度. 根据经验和实践，按如下方法选取是比较好的，即初始向量的第 1 列的元素全部置为 1，以后各列按 m_{ii}/k_{ii} 中的大小排序，分别在该顺序对应的位置上置 $1/m_{kk}$，而该列其余元素皆置 0，其中 k 即是指第 k 个元素最大.

子空间迭代法的基本流程如下：

选定初始向量

$$\hat{X}_1 = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_m^{(1)} \end{bmatrix}$$

进行迭代运算

$$Y_k = M\hat{X}_k, \quad KX_{k+1} = Y_k$$

进行 Ritz 方法运算

$$\hat{X}_{k+1} = X_{k+1} P_{k+1}$$

求解新的降阶的广义特征值问题

$$K_{k+1} P_{k+1} = M_{k+1} P_{k+1} \lambda_{k+1}$$

其中，$M_{k+1} = X_{k+1}^{\mathrm{T}} M X_{k+1}$，$K_{k+1} = X_{k+1}^{\mathrm{T}} K X_{k+1}$.

如果特征值的精度满足要求，则终止循环. 计算原特征值问题的特征值和特征向量为

$$\lambda_i \approx \lambda_i^{*}, \quad \psi_i \approx X_{k+1} P_{k+1,i}$$

MATLAB 提供了专门的函数来计算矩阵的特征值，即 eig 和 eigs，它们既可用于求解标准特征值问题，也可用于求解广义特征值问题. 前者一般用来求解维数较小的全部特征值，而后者一般用来获取大型特征值问题的某几个特征对. 其调用格式如下：

```
E = eig(K, M)
[V, D] = eig(K, M);
[V, D] = eig(K, M, flag)        flag ='chol' or 'qz'
[V, D]=eigs(K, M, nummode, SIGMA)        SIGMA='SM' or 'LM'
```

2.2 时域动力响应分析

2.2.1 数值积分法

求解结构动力学方程的一种方法是直接积分法. 常用的方法有中心差分法、Houbolt 法、Wilson - θ 法和 Newmark 法.

直接积分法的基本思路是指在相隔 Δt 的一些离散点上而不是在任何时刻 t 上满足运动方程，并且在每一个时间区域内假定位移、速度和加速度的变化规律来得到运动方程的解.

对于有限元法建立的动力学方程

$$M\ddot{\delta} + C\dot{\delta} + K\delta = F \tag{2-13}$$

假定在时间 $t = 0$ 时的位移 δ_0、速度 $\dot{\delta}_0$ 和 $\ddot{\delta}_0$ 是已知的.

1. 中心差分法

在中心差分法中，加速度和速度用位移表示为

$$\ddot{\boldsymbol{\delta}}_t=\frac{1}{\Delta t}\left(\frac{\boldsymbol{\delta}_{t+\Delta t}-\boldsymbol{\delta}_t}{\Delta t}-\frac{\boldsymbol{\delta}_t-\boldsymbol{\delta}_{t-\Delta t}}{\Delta t}\right)=\frac{1}{\Delta t^2}(\boldsymbol{\delta}_{t-\Delta t}-2\boldsymbol{\delta}_t+\boldsymbol{\delta}_{t+\Delta t}) \tag{2-14}$$

$$\dot{\boldsymbol{\delta}}_t=\frac{1}{2\Delta t}(-\boldsymbol{\delta}_{t-\Delta t}+\boldsymbol{\delta}_{t+\Delta t}) \tag{2-15}$$

将式(2-14)和式(2-15)代入式(2-13)，可得到由方程 t 时刻的解表示的位移

$$\left(\frac{1}{\Delta t^2}\boldsymbol{M}+\frac{1}{2\Delta t}\boldsymbol{C}\right)\boldsymbol{\delta}_{t+\Delta t}=\boldsymbol{F}_t-\left(\boldsymbol{K}-\frac{2}{\Delta t^2}\boldsymbol{M}\right)\boldsymbol{\delta}_t-\left(\frac{1}{\Delta t^2}\boldsymbol{M}-\frac{1}{2\Delta t}\boldsymbol{C}\right)\boldsymbol{\delta}_{t-\Delta t} \tag{2-16}$$

在开始计算时，式(2-16)中 $\boldsymbol{\delta}_{-\Delta t}$ 需要利用初始条件 $\boldsymbol{\delta}_0$ 和 $\dot{\boldsymbol{\delta}}_0$ 计算. 由式(2-14)、式(2-15)和式(2-16)，有

$$\boldsymbol{\delta}_{-\Delta t}=\boldsymbol{\delta}_0-\Delta t\dot{\boldsymbol{\delta}}_0+\frac{\Delta t^2}{2}\ddot{\boldsymbol{\delta}}_0 \tag{2-17}$$

$$\ddot{\boldsymbol{\delta}}_0=\boldsymbol{M}^{-1}(\boldsymbol{F}_0-\boldsymbol{C}\dot{\boldsymbol{\delta}}_0-\boldsymbol{K}\boldsymbol{\delta}_0) \tag{2-18}$$

中心差分法是一种显式积分格式，算法是条件稳定的，稳定条件为

$$\Delta t\leqslant\Delta t_{cr}=\frac{T_n}{\pi} \tag{2-19}$$

式中，T_n 是结构系统的最小固有振动周期.

中心差分法的计算步骤可归结如下.

1) 初始计算

(1) 形成刚度矩阵 $\boldsymbol{K}$、质量矩阵 $\boldsymbol{M}$ 和阻尼矩阵 $\boldsymbol{C}$.

(2) 由初始条件 $\boldsymbol{\delta}_0$ 和 $\dot{\boldsymbol{\delta}}_0$，计算 $\ddot{\boldsymbol{\delta}}_0$.

$$\ddot{\boldsymbol{\delta}}_0=\boldsymbol{M}^{-1}(\boldsymbol{F}_0-\boldsymbol{C}\dot{\boldsymbol{\delta}}_0-\boldsymbol{K}\boldsymbol{\delta}_0)$$

(3) 选取时间步长 Δt，$\Delta t<\Delta t_{cr}$，计算积分常数.

$$A_0=\Delta t,\quad A_1=\frac{\Delta t^2}{2},\quad A_2=\frac{1}{\Delta t^2},\quad A_3=\frac{1}{2\Delta t}$$

(4) 计算 $\boldsymbol{\delta}_{-\Delta t}$.

$$\boldsymbol{\delta}_{-\Delta t}=\boldsymbol{\delta}_0-A_0\dot{\boldsymbol{\delta}}_0+A_1\ddot{\boldsymbol{\delta}}_0$$

(5) 计算有效质量矩阵 $\bar{\boldsymbol{M}}$.

$$\bar{\boldsymbol{M}}=A_2\boldsymbol{M}+A_3\boldsymbol{C}$$

2) 对于每一时间步长

(1) 计算 t 时刻的有效载荷向量 $\bar{\boldsymbol{F}}$.

$$\bar{\boldsymbol{F}}_t=\boldsymbol{F}_t-(\boldsymbol{K}-2A_2\boldsymbol{M})\boldsymbol{\delta}_t-(A_2\boldsymbol{M}-A_3\boldsymbol{C})\ \boldsymbol{\delta}_{t-\Delta t}$$

(2) 求解 $t+\Delta t$ 时刻的位移.

$$\boldsymbol{\delta}_{t+\Delta t}=\bar{\boldsymbol{M}}^{-1}\bar{\boldsymbol{F}}_t$$

(3) 计算 t 时刻的加速度和速度.

$$\ddot{\boldsymbol{\delta}}_t=A_2(\boldsymbol{\delta}_{t-\Delta t}-2\boldsymbol{\delta}_t+\boldsymbol{\delta}_{t+\Delta t})$$

$$\dot{\boldsymbol{\delta}}_t=A_3(-\boldsymbol{\delta}_{t-\Delta t}+\boldsymbol{\delta}_{t+\Delta t})$$

2. Houbolt 法

Houbolt 的积分格式也是一种有限差分格式，加速度和速度的有限差分展开式为

$$\ddot{\boldsymbol{\delta}}_{t+\Delta t}=\frac{1}{\Delta t^2}(2\boldsymbol{\delta}_{t+\Delta t}-5\boldsymbol{\delta}_t+4\boldsymbol{\delta}_{t-\Delta t}-\boldsymbol{\delta}_{t-2\Delta t}) \tag{2-20}$$

$$\dot{\boldsymbol{\delta}}_{t+\Delta t}=\frac{1}{6\Delta t}(11\boldsymbol{\delta}_{t+\Delta t}-18\boldsymbol{\delta}_t+9\boldsymbol{\delta}_{t-\Delta t}-2\boldsymbol{\delta}_{t-2\Delta t}) \tag{2-21}$$

在 $t+\Delta t$ 时刻，式(2-13)可表示为

$$\boldsymbol{M}\ddot{\boldsymbol{\delta}}_{t+\Delta t}+\boldsymbol{C}\dot{\boldsymbol{\delta}}_{t+\Delta t}+\boldsymbol{K}\boldsymbol{\delta}_{t+\Delta t}=\boldsymbol{F}_{t+\Delta t} \tag{2-22}$$

将式(2-20)和式(2-21)代入式(2-22)，可得

$$\left(\frac{2}{\Delta t^2}\boldsymbol{M}+\frac{11}{6\Delta t}\boldsymbol{C}+\boldsymbol{K}\right)\boldsymbol{\delta}_{t+\Delta t}=\boldsymbol{F}_{t+\Delta t}+\left(\frac{5}{\Delta t^2}\boldsymbol{M}+\frac{3}{\Delta t}\boldsymbol{C}\right)\boldsymbol{\delta}_t-\left(\frac{4}{\Delta t^2}\boldsymbol{M}+\frac{3}{2\Delta t}\boldsymbol{C}\right)\boldsymbol{\delta}_{t-\Delta t}+\left(\frac{1}{\Delta t^2}\boldsymbol{M}+\frac{1}{3\Delta t}\boldsymbol{C}\right)\boldsymbol{\delta}_{t-2\Delta t} \tag{2-23}$$

在式(2-23)中，求解 $\boldsymbol{\delta}_{t+\Delta t}$ 需要已知 $\boldsymbol{\delta}_t$、$\boldsymbol{\delta}_{t-\Delta t}$ 和 $\boldsymbol{\delta}_{t-2\Delta t}$. 虽然由初始条件可知 $\boldsymbol{\delta}_0$、$\dot{\boldsymbol{\delta}}_0$ 和 $\ddot{\boldsymbol{\delta}}_0$，但是 $\boldsymbol{\delta}_{\Delta t}$ 和 $\boldsymbol{\delta}_{2\Delta t}$ 需要用其他计算方法确定.

Houbolt 法是一种隐式积分格式，算法是无条件稳定的.

Houbolt 法的计算步骤可归结如下.

1) 初始计算

(1) 形成刚度矩阵 $\boldsymbol{K}$、质量矩阵 $\boldsymbol{M}$ 和阻尼矩阵 $\boldsymbol{C}$.

(2) 由初始条件 $\boldsymbol{\delta}_0$ 和 $\dot{\boldsymbol{\delta}}_0$，计算 $\ddot{\boldsymbol{\delta}}_0$.

$$\ddot{\boldsymbol{\delta}}_0=\boldsymbol{M}^{-1}(\boldsymbol{F}_0-\boldsymbol{C}\dot{\boldsymbol{\delta}}_0-\boldsymbol{K}\boldsymbol{\delta}_0)$$

(3) 选取时间步长 Δt，计算积分常数.

$$A_0=\Delta t,\quad A_1=\frac{\Delta t^2}{2},\quad A_2=\frac{1}{\Delta t^2},\quad A_3=\frac{1}{6\Delta t}$$

(4) 用其他方法计算 $\boldsymbol{\delta}_{\Delta t}$ 和 $\boldsymbol{\delta}_{2\Delta t}$.

(5) 计算有效刚度矩阵 $\bar{\boldsymbol{K}}$.

$$\bar{\boldsymbol{K}}=\boldsymbol{K}+2A_2\boldsymbol{M}+11A_3\boldsymbol{C}$$

2) 对于每一时间步长

(1) 计算 $t+\Delta t$ 时刻的有效载荷向量 $\bar{\boldsymbol{F}}$.

$$\bar{\boldsymbol{F}}_{t+\Delta t}=\boldsymbol{F}_{t+\Delta t}+[5A_2\boldsymbol{\delta}_t-4A_2\boldsymbol{\delta}_{t-\Delta t}+A_2\boldsymbol{\delta}_{t-2\Delta t}]\boldsymbol{M}+[3A_3\boldsymbol{\delta}_t-9A_3\boldsymbol{\delta}_{t-\Delta t}+2A_3\boldsymbol{\delta}_{t-2\Delta t}]\boldsymbol{C}$$

(2) 求解 $t+\Delta t$ 时刻的位移.

$$\boldsymbol{\delta}_{t+\Delta t}=\bar{\boldsymbol{K}}^{-1}\bar{\boldsymbol{F}}_{t+\Delta t}$$

(3) 计算 $t+\Delta t$ 时刻的加速度和速度.

$$\ddot{\boldsymbol{\delta}}_{t+\Delta t}=A_2(2\boldsymbol{\delta}_{t+\Delta t}-5\boldsymbol{\delta}_t+4\boldsymbol{\delta}_{t-\Delta t}-\boldsymbol{\delta}_{t-2\Delta t})$$

$$\dot{\boldsymbol{\delta}}_{t+\Delta t}=A_3(11\boldsymbol{\delta}_{t+\Delta t}-18\boldsymbol{\delta}_t+9\boldsymbol{\delta}_{t-\Delta t}-2\boldsymbol{\delta}_{t-2\Delta t})$$

3. Wilson - θ 法

Wilson-θ 法是线性加速度法的推广. Wilson - θ 法假定加速度从时刻 t 到时刻 $t+\theta\Delta t$ 为

线性变化，其中 $\theta \geqslant 1$.

若令时间增量为 τ，且 $0 \leqslant \tau \leqslant \theta\Delta t$，则由线性假设，在 t 到 $t+\theta\Delta t$ 的时间区间内，有

$$\ddot{\boldsymbol{\delta}}_{t+\tau} = \ddot{\boldsymbol{\delta}}_t + \frac{\tau}{\theta\Delta t}(\ddot{\boldsymbol{\delta}}_{t+\theta\Delta t} - \ddot{\boldsymbol{\delta}}_t) \tag{2-24}$$

对式(2-24)积分，得

$$\dot{\boldsymbol{\delta}}_{t+\tau} = \dot{\boldsymbol{\delta}}_t + \ddot{\boldsymbol{\delta}}_t\tau + \frac{\tau^2}{2\theta\Delta t}(\ddot{\boldsymbol{\delta}}_{t+\theta\Delta t} - \ddot{\boldsymbol{\delta}}_t) \tag{2-25}$$

$$\boldsymbol{\delta}_{t+\tau} = \boldsymbol{\delta}_t + \dot{\boldsymbol{\delta}}_t\tau + \frac{1}{2}\ddot{\boldsymbol{\delta}}_t\tau^2 + \frac{\tau^3}{6\theta\Delta t}(\ddot{\boldsymbol{\delta}}_{t+\theta\Delta t} - \ddot{\boldsymbol{\delta}}_t) \tag{2-26}$$

于是，在时刻 $t+\theta\Delta t$，有

$$\dot{\boldsymbol{\delta}}_{t+\theta\Delta t} = \dot{\boldsymbol{\delta}}_t + \frac{\theta\Delta t}{2}(\ddot{\boldsymbol{\delta}}_{t+\theta\Delta t} + \ddot{\boldsymbol{\delta}}_t) \tag{2-27}$$

$$\boldsymbol{\delta}_{t+\theta\Delta t} = \boldsymbol{\delta}_t + \theta\Delta t\dot{\boldsymbol{\delta}}_t + \frac{\theta^2\Delta t^2}{6}(\ddot{\boldsymbol{\delta}}_{t+\theta\Delta t} + 2\ddot{\boldsymbol{\delta}}_t) \tag{2-28}$$

联解式(2-27)和式(2-28)，可得到时刻 $t+\theta\Delta t$ 的加速度和速度

$$\ddot{\boldsymbol{\delta}}_{t+\theta\Delta t} = \frac{6}{\theta^2\Delta t^2}(\boldsymbol{\delta}_{t+\theta\Delta t} - \boldsymbol{\delta}_t) - \frac{6}{\theta\Delta t}(\dot{\boldsymbol{\delta}}_t - 2\ddot{\boldsymbol{\delta}}_t) \tag{2-29}$$

$$\dot{\boldsymbol{\delta}}_{t+\theta\Delta t} = \frac{3}{\theta\Delta t}(\boldsymbol{\delta}_{t+\theta\Delta t} - \boldsymbol{\delta}_t) - 2\dot{\boldsymbol{\delta}}_t - \frac{\theta\Delta t}{2}\ddot{\boldsymbol{\delta}}_t \tag{2-30}$$

同样考察在时刻 $t+\theta\Delta t$ 的运动方程，式(2-13)可表示为

$$\boldsymbol{M}\ddot{\boldsymbol{\delta}}_{t+\theta\Delta t} + \boldsymbol{C}\dot{\boldsymbol{\delta}}_{t+\theta\Delta t} + \boldsymbol{K}\boldsymbol{\delta}_{t+\theta\Delta t} = \boldsymbol{F}_{t+\theta\Delta t} \tag{2-31}$$

将式(2-29)和式(2-30)代入式(2-31)，可得

$$\left(\frac{6}{\theta^2\Delta t^2}\boldsymbol{M} + \frac{3}{\theta\Delta t}\boldsymbol{C} + \boldsymbol{K}\right)\boldsymbol{\delta}_{t+\theta\Delta t} = \boldsymbol{F}_{t+\theta\Delta t} + \left(2\boldsymbol{M} + \frac{\theta\Delta t}{2}\boldsymbol{C}\right)\ddot{\boldsymbol{\delta}}_t + \left(\frac{6}{\theta\Delta t}\boldsymbol{M} + 2\boldsymbol{C}\right)\dot{\boldsymbol{\delta}}_t + \left(\frac{6}{\theta^2\Delta t^2}\boldsymbol{M} + \frac{3}{\theta\Delta t}\boldsymbol{C}\right)\boldsymbol{\delta}_t \tag{2-32}$$

由式(2-32)求解出位移 $\boldsymbol{\delta}_{t+\theta\Delta t}$，代入式(2-29)即可计算 $\ddot{\boldsymbol{\delta}}_{t+\theta\Delta t}$，再代入式(2-24)、式(2-25)和式(2-26)，并令 $\tau = \Delta t$，则有

$$\ddot{\boldsymbol{\delta}}_{t+\Delta t} = \frac{6}{\theta^3\Delta t^2}(\boldsymbol{\delta}_{t+\theta\Delta t} - \boldsymbol{\delta}_t) - \frac{6}{\theta^2\Delta t}\dot{\boldsymbol{\delta}}_t + (1 - \frac{3}{\theta})\ddot{\boldsymbol{\delta}}_t \tag{2-33}$$

$$\dot{\boldsymbol{\delta}}_{t+\Delta t} = \dot{\boldsymbol{\delta}}_t + \ddot{\boldsymbol{\delta}}_t\Delta t + \frac{\Delta t}{2}(\ddot{\boldsymbol{\delta}}_{t+\Delta t} - \ddot{\boldsymbol{\delta}}_t) \tag{2-34}$$

$$\boldsymbol{\delta}_{t+\Delta t} = \boldsymbol{\delta}_t + \dot{\boldsymbol{\delta}}_t\Delta t + \frac{1}{2}\ddot{\boldsymbol{\delta}}_t\Delta t^2 + \frac{\Delta t^2}{6}(\ddot{\boldsymbol{\delta}}_{t+\Delta t} - \ddot{\boldsymbol{\delta}}_t) \tag{2-35}$$

Wilson-θ 法是一种隐式积分方法，当 $\theta \geqslant 1.37$ 时，该算法无条件稳定，通常计算时取 $\theta = 1.4$.

Wilson-θ 法的计算步骤可归结如下.

1) 初始计算

(1) 形成刚度矩阵 $\boldsymbol{K}$、质量矩阵 $\boldsymbol{M}$ 和阻尼矩阵 $\boldsymbol{C}$.

(2) 由初始条件 δ_0 和 $\dot{\delta}_0$，计算 $\ddot{\delta}_0$.

$$\ddot{\delta}_0 = M^{-1}(F_0 - C\dot{\delta}_0 - K\delta_0)$$

(3) 选取时间步长 Δt，计算积分常数，取 $\theta = 1.4$.

$$A_1 = \frac{6}{\theta^2 \Delta t^2},\quad A_2 = \frac{3}{\theta \Delta t},\quad A_3 = \frac{\theta \Delta t}{2}$$

(4) 计算有效刚度矩阵 $\bar{K}$.

$$\bar{K} = K + A_1 M + A_2 C$$

2) 对于每一时间步长

(1) 计算 $t+\theta\Delta t$ 时刻的有效载荷向量 $\bar{F}$.

$$\bar{F}_{t+\theta\Delta t} = F_{t+\theta\Delta t} + [A_1\delta_t + 2A_2\dot{\delta}_t + 2\ddot{\delta}_t]M + [A_2\delta_t + 2\dot{\delta}_t + A_3\ddot{\delta}_t]C$$

(2) 求解 $t+\theta\Delta t$ 时刻的位移.

$$\delta_{t+\theta\Delta t} = \bar{K}^{-1}\bar{F}_{t+\theta\Delta t}$$

(3) 计算 $t+\Delta t$ 时刻的加速度、速度和位移.

$$\ddot{\delta}_{t+\Delta t} = \frac{6}{\theta^3 \Delta t^2}(\delta_{t+\theta\Delta t} - \delta_t) - \frac{6}{\theta^2 \Delta t}\dot{\delta}_t + (1 - \frac{3}{\theta})\ddot{\delta}_t$$

$$\dot{\delta}_{t+\Delta t} = \dot{\delta}_t + \frac{\Delta t}{2}(\ddot{\delta}_{t+\Delta t} + \ddot{\delta}_t)$$

$$\delta_{t+\Delta t} = \delta_t + \dot{\delta}_t\Delta t + \frac{\Delta t^2}{6}(\ddot{\delta}_{t+\Delta t} + 2\ddot{\delta}_t)$$

4. Newmark 法

Newmark 积分格式也可以看成是线性加速度方法的推广. Newmark 法所采用的假设为

$$\dot{\delta}_{t+\Delta t} = \dot{\delta}_t + [(1-\beta)\ddot{\delta}_t + \beta\ddot{\delta}_{t+\Delta t}]\Delta t \tag{2-36}$$

$$\delta_{t+\Delta t} = \delta_t + \dot{\delta}_t\Delta t + [(\frac{1}{2} - \alpha)\ddot{\delta}_t + \alpha\ddot{\delta}_{t+\Delta t}]\Delta t^2 \tag{2-37}$$

式中，参数 α 和 β 根据积分的精度和稳定性要求来确定. 当 $\alpha = 1/6$，$\beta = 1/2$ 时，即为线性加速度方法. Newmark 法取 $\alpha = 1/4$，$\beta = 1/2$，是一种平均加速度方法.

由式(2-36)和式(2-37)可解出

$$\ddot{\delta}_{t+\Delta t} = \frac{1}{\alpha\Delta t^2}(\delta_{t+\Delta t} - \delta_t) - \frac{1}{\alpha\Delta t}\dot{\delta}_t - (\frac{1}{2\alpha} - 1)\ddot{\delta}_t \tag{2-38}$$

$$\dot{\delta}_{t+\Delta t} = \dot{\delta}_t + (1-\beta)\Delta t\ddot{\delta}_t + \beta\Delta t\ddot{\delta}_{t+\Delta t} \tag{2-39}$$

将式(2-38)和式(2-39)代入时刻 $t+\Delta t$ 的运动方程

$$M\ddot{\delta}_{t+\Delta t} + C\dot{\delta}_{t+\Delta t} + K\delta_{t+\Delta t} = F_{t+\Delta t}$$

得到

$$\begin{aligned}(\frac{1}{\alpha\Delta t^2}M + \frac{\beta}{\alpha\Delta t}C + K)\delta_{t+\Delta t} = {} & F_{t+\Delta t} + [(\frac{1}{2\alpha} - 1)M + \frac{\Delta t}{2}(\frac{\beta}{\alpha} - 2)C]\ddot{\delta}_t \\ & + [\frac{1}{\alpha\Delta t}M + (\frac{\beta}{\alpha} - 1)C]\dot{\delta}_t + (\frac{1}{\alpha\Delta t^2}M + \frac{\beta}{\alpha\Delta t}C)\delta_t\end{aligned} \tag{2-40}$$

求解式(2-40)得到$\delta_{t+\Delta t}$，再由式(2-38)和式(2-39)计算$\ddot{\delta}_{t+\Delta t}$和$\dot{\delta}_{t+\Delta t}$.

Newmark 法是一种隐式积分方法，当$\beta \geqslant 0.50$和$\alpha \geqslant 0.25(0.5+\beta)^2$时，该算法无条件稳定.

Newmark 法的计算步骤可归结如下.

1) 初始计算

(1) 形成刚度矩阵$\boldsymbol{K}$、质量矩阵$\boldsymbol{M}$和阻尼矩阵$\boldsymbol{C}$.

(2) 由初始条件$\boldsymbol{\delta}_0$和$\dot{\boldsymbol{\delta}}_0$，计算$\ddot{\boldsymbol{\delta}}_0$.

$$\ddot{\boldsymbol{\delta}}_0 = \boldsymbol{M}^{-1}(\boldsymbol{F}_0 - \boldsymbol{C}\dot{\boldsymbol{\delta}}_0 - \boldsymbol{K}\boldsymbol{\delta}_0)$$

(3) 选取时间步长Δt和参数α、β，计算积分常数.

$\beta \geqslant 0.50$，　$\alpha \geqslant 0.25(0.5+\beta)^2$

$$A_1 = \frac{1}{\alpha \Delta t^2},\quad A_2 = \frac{\beta}{\alpha \Delta t},\quad A_3 = \frac{1}{\alpha \Delta t}$$

$$A_4 = \left(\frac{1}{2\alpha} - 1\right),\quad A_5 = \frac{\Delta t}{2}\left(\frac{\beta}{\alpha} - 2\right),\quad A_6 = \left(\frac{\beta}{\alpha} - 1\right)$$

(4) 计算有效刚度矩阵$\bar{\boldsymbol{K}}$.

$$\bar{\boldsymbol{K}} = \boldsymbol{K} + A_1\boldsymbol{M} + A_2\boldsymbol{C}$$

2) 对于每一时间步长

(1) 计算$t+\Delta t$时刻的有效载荷向量$\bar{\boldsymbol{F}}$.

$$\bar{\boldsymbol{F}}_{t+\Delta t} = \boldsymbol{F}_{t+\Delta t} + [A_1\boldsymbol{\delta}_t + A_3\dot{\boldsymbol{\delta}}_t + A_4\ddot{\boldsymbol{\delta}}_t]\boldsymbol{M} + [A_2\boldsymbol{\delta}_t + A_6\dot{\boldsymbol{\delta}}_t + A_5\ddot{\boldsymbol{\delta}}_t]\boldsymbol{C}$$

(2) 求解$t+\Delta t$时刻的位移.

$$\boldsymbol{\delta}_{t+\Delta t} = \bar{\boldsymbol{K}}^{-1}\bar{\boldsymbol{F}}_{t+\Delta t}$$

(3) 计算$t+\Delta t$时刻的加速度、速度.

$$\ddot{\boldsymbol{\delta}}_{t+\Delta t} = \frac{1}{\alpha \Delta t^2}(\boldsymbol{\delta}_{t+\Delta t} - \boldsymbol{\delta}_t) - \frac{1}{\alpha \Delta t}\dot{\boldsymbol{\delta}}_t - \left(\frac{1}{2\alpha} - 1\right)\ddot{\boldsymbol{\delta}}_t$$

$$\dot{\boldsymbol{\delta}}_{t+\Delta t} = \dot{\boldsymbol{\delta}}_t + (1-\beta)\Delta t\ddot{\boldsymbol{\delta}}_t + \beta\Delta t\ddot{\boldsymbol{\delta}}_{t+\Delta t}$$

5. 状态空间法

式(2-13)在状态空间的表达式为

$$\dot{\boldsymbol{X}} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{B} \tag{2-41}$$

其中

$$\boldsymbol{X} = \begin{Bmatrix} \boldsymbol{\delta} \\ \dot{\boldsymbol{\delta}} \end{Bmatrix},\quad \boldsymbol{A} = \begin{bmatrix} 0 & \boldsymbol{I} \\ -\boldsymbol{M}^{-1}\boldsymbol{K} & -\boldsymbol{M}^{-1}\boldsymbol{C} \end{bmatrix},\quad \boldsymbol{B} = \begin{bmatrix} 0 \\ \boldsymbol{M}^{-1}\boldsymbol{F} \end{bmatrix} \tag{2-42}$$

进一步结合系统的输出方程，系统方程可表示为

$$\begin{cases} \dot{\boldsymbol{X}} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{B} \\ \boldsymbol{Y} = \boldsymbol{C}_T\boldsymbol{X} \end{cases} \tag{2-43}$$

控制系统最常用的时域分析方法是，当输入信号为单位阶跃和单位脉冲(冲击)函数时，求出系统的输出响应，分别称为单位阶跃响应和单位冲击响应. 在 MATLAB 中，提

供了求解连续系统的单位阶跃响应函数 step()单位冲击响应函数 impulse()，零输入响应函数 initial()及任意输入下的仿真函数 lsim().

1) 阶跃响应分析

控制系统工具箱中给出了 step()函数来直接求解线性系统的阶跃响应，其调用格式为

y=step(G, t)：变量 t 为由要计算点所在时刻的值组成的向量.

[y, t]=step(G)：时间向量 t 由系统模型 G 自动生成.

[y, t, x]=step(G)：时间向量 t 由系统模型 G 自动生成，并得出状态变量响应.

step(G)：自动绘制系统的阶跃响应曲线.

其中 G 为给定控制系统的 LTI 对象模型.

2) 脉冲响应分析

控制系统工具箱中给出了 impulse()函数来直接求解线性系统的脉冲响应，其调用格式为

y=impulse(G, t)：变量 t 为由要计算点所在时刻的值组成的向量.

[y, t]=impulse(G)：时间向量 t 由系统模型 G 自动生成.

[y, t, x]=impulse(G)：时间向量 t 由系统模型 G 自动生成，并得出状态变量响应.

impulse(G)：自动绘制系统的脉冲响应曲线.

3) 任意输入下的时域响应分析

(1) 零输入响应：输入 $u(t)=0$，则 $\hat{x}(t)=e^{At}x(0)$.

控制系统工具箱中给出了 initial( )函数来求取系统的零输入响应，其调用格式为

y=initial(G, X0)

[y, t, x]=initial(G, X0)

[y, t, x]=initial(G, X0, t)

其中 X0 为系统的初始状态变量.

(2) 零状态响应：系统的初始状态 $x(0)=0$，则 $\hat{x}(t)=\int_0^t e^{A(t-\tau)}Bu(\tau)d\tau$.

控制系统工具箱中给出了 lsim( )函数来求取系统的零状态响应，其调用格式为

y=lsim(G, u, t)

(3) 一般响应.

同样，可以由 lsim()函数求得既有输入且初始状态不为零的情况下系统的响应，其调用格式为

[y, t, x]=lsim(G, u, t, X0)

其中要求 t 是一个间隔均匀的时间值列向量，输入 u 的列数同输入变量的数目相同，行数同 t 的行数相同. 状态响应用 x 返回，它有 n 列，行数与 t 相同.

2.2.2　振型叠加法

1. 无阻尼系统的动力学响应

对于式(2-13)的自由振动方程为

$$M\ddot{\delta}+K\delta=0 \tag{2-44}$$

可假设其解的形式成如下的指数形式

$$\delta(t)=\phi \mathrm{e}^{\mathrm{i}\omega t} \tag{2-45}$$

式中，ϕ 和 ω 分别为运动的振型和固有频率．将式(2-45)代入式(2-44)，可得

$$(-\omega^2 M+K)\phi \mathrm{e}^{\mathrm{i}\omega t}=0 \tag{2-46}$$

式(2-46)方程具有非平凡解的条件是 $(-\omega^2[M]+[K])$ 奇异，即

$$\left|-\omega^2 M+K\right|=\left|-\lambda M+K\right|=0 \tag{2-47}$$

式中，$\lambda=\omega^2$．求解式(2-47)可得 n 个特征值 $\omega_1^2,\omega_2^2,\cdots,\omega_n^2$ 和特征向量 $\phi_1,\phi_2,\cdots,\phi_n$．

对于任意两个特征解 (ω_i^2,ϕ_i)、(ω_j^2,ϕ_j)，由式(2-46)，有

$$K\phi_i=\omega_i^2 M\phi_i,\quad K\phi_j=\omega_j^2 M\phi_j \tag{2-48}$$

上述方程的第一式两边左乘 ϕ_j^{T}，第二式左乘 ϕ_i^{T} 并利用矩阵 K 和 M 的对称性进行转置，得

$$\phi_j^{\mathrm{T}}K\phi_i=\omega_i^2\phi_j^{\mathrm{T}}M\phi_i,\quad \phi_j^{\mathrm{T}}K\phi_i=\omega_j^2\phi_j^{\mathrm{T}}M\phi_i \tag{2-49}$$

将两式相减得到

$$(\omega_i^2-\omega_j^2)\phi_j^{\mathrm{T}}M\phi_i=0 \tag{2-50}$$

于是，可得到振型关于质量矩阵和刚度矩阵的正交性

$$\phi_j^{\mathrm{T}}M\phi_i=\begin{cases}m_i & (i=j)\\ 0 & (i\neq j)\end{cases} \tag{2-51}$$

$$\phi_j^{\mathrm{T}}K\phi_i=\begin{cases}k_i & (i=j)\\ 0 & (i\neq j)\end{cases} \tag{2-52}$$

再利用振型关于质量矩阵的正则化，有

$$\phi_j^{\mathrm{T}}M\phi_i=\begin{cases}1 & (i=j)\\ 0 & (i\neq j)\end{cases} \tag{2-53}$$

$$\phi_j^{\mathrm{T}}K\phi_i=\begin{cases}\omega_i & (i=j)\\ 0 & (i\neq j)\end{cases} \tag{2-54}$$

利用特征向量，可以定义一个坐标变换

$$\delta=\Phi\eta \tag{2-55}$$

其中

$$\Phi=[\phi_1,\phi_2,\cdots,\phi_n] \tag{2-56}$$

$$\eta=\{\eta_1,\eta_2,\cdots,\eta_n\}^{\mathrm{T}} \tag{2-57}$$

分别称为模态矩阵和模态坐标向量．

将式(2-55)代入相应于式(2-13)的无阻尼方程中，可得

$$M\Phi\ddot{\eta}+K\Phi\eta=F \tag{2-58}$$

方程两端左乘 Φ^{T}，则有

$$\Phi^{\mathrm{T}}M\Phi\ddot{\eta}+\Phi^{\mathrm{T}}K\Phi\eta=\Phi^{\mathrm{T}}F \tag{2-59}$$

由式(2-53)和式(2-54)，式(2-59)变为

$$\ddot{\eta}+\Omega^2\eta=Q \tag{2-60}$$

其中

$$\Omega^2=\mathrm{diag}[\omega_1^2\quad\omega_2^2\quad\cdots\quad\omega_n^2],\quad Q=\Phi^{\mathrm{T}}F \tag{2-61}$$

分别称为固有频率矩阵和模态力矩阵.

式(2-59)为一组解耦的方程组，也可写为

$$\ddot{\eta}_i+\omega_i^2\eta_i=f_i\quad i=1,2,\cdots,n \tag{2-62}$$

式中，f_i为模态力矩阵的第i行.

对于式(2-62)的解可由Duhamel积分得到

$$\eta_i(t)=\eta_i(t_0)\cos\omega_i t+\frac{\dot{\eta}_i(t_0)}{\omega_i}\sin\omega_i t+\frac{1}{\omega_i}\int_{t_0}^{t}\sin\omega_i(t-\tau)f_i(\tau)\mathrm{d}\tau \tag{2-63}$$

当激励力f_i为脉冲、阶跃和简谐函数时，上述方程中的卷积积分可直接求出. 对于脉冲激励$f_i(t)=F_{0i}\delta(t)$，$t_0=0$，式(2-63)的解为

$$\eta_i(t)=\eta_i(0)\cos\omega_i t+\frac{\dot{\eta}_i(0)}{\omega_i}\sin\omega_i t+\frac{1}{\omega_i}F_{0i}\sin\omega_i t$$

对于阶跃激励$f_i(t)=F_{0i}$，$t_0=0$，式(2-63)的解为

$$\eta_i(t)=\eta_i(0)\cos\omega_i t+\frac{\dot{\eta}_i(0)}{\omega_i}\sin\omega_i t+\frac{1}{\omega_i^2}F_{0i}(1-\cos\omega_i t)$$

对于简谐激励$f_i(t)=F_{0i}\cos\omega t$，$t_0=0$，式(2-63)的解为

$$\eta_i(t)=\eta_i(0)\cos\omega_i t+\frac{\dot{\eta}_i(0)}{\omega_i}\sin\omega_i t+\frac{F_{0i}}{\omega_i^2}\frac{1}{1-\gamma^2}(\cos\omega t-\cos\omega_i t)$$

式中，$\gamma=\omega/\omega_i$.

对应于式(2-63)的初始条件$\eta_i(0)$、$\dot{\eta}_i(0)$可由原坐标系下的初始条件变换得到. 由式(2-55)，原坐标系下的初始条件可表示为

$$\delta(0)=\Phi\eta(0),\quad \dot{\delta}(0)=\Phi\dot{\eta}(0) \tag{2-64}$$

因而有

$$\eta(0)=\Phi^{\mathrm{T}}M\delta(0),\quad \dot{\eta}(0)=\Phi^{\mathrm{T}}M\dot{\delta}(0) \tag{2-65}$$

当模态坐标下的响应求出后，即可叠加得到实际坐标系下的响应

$$\delta(t)=\Phi\eta(t) \tag{2-66}$$

或

$$\delta_i(t)=\sum_{j=1}^{n}\phi_{ij}\eta_j(t)\quad(i=1,2,\cdots,n) \tag{2-67}$$

2. 有阻尼系统的动力学响应

对于有阻尼的结构系统，运动方程为

$$M\ddot{\delta}+C\dot{\delta}+K\delta=F \tag{2-68}$$

式中，C为阻尼矩阵.

对于阻尼力正比于结构运动速度的假设，阻尼矩阵的计算式为

$$C^e = \int_{V_e} cN^T N dv \tag{2-69}$$

式中，c 为阻尼系数，N 为形函数.

结构的阻尼一般是难以准确确定的，实际应用中常以实测数据由振动过程中结构系统的能量消耗来给出阻尼的近似值，因此阻尼矩阵一般不是直接计算单元的阻尼矩阵，而是直接给出结构的总体阻尼矩阵. 实际工程结构的阻尼多是较小的数值，一般采用线性关系

$$C = \alpha M + \beta K \tag{2-70}$$

称为比例阻尼或 Rayleigh 阻尼. 式中，α 和 β 为常数.

比例阻尼的优点是模态向量关于质量矩阵和刚度矩阵的正交性对于比例阻尼仍成立，即有

$$\phi_j^T C \phi_i = \begin{cases} c_i & i = j \\ 0 & i \neq j \end{cases} \tag{2-71}$$

$\phi_j^T C \phi_i$ 为一对角矩阵.

对式(2-68)仍用模态矩阵进行变换，方程变为

$$\ddot{\eta}_i + 2\xi_i \omega_i \dot{\eta}_i + \omega_i^2 \eta_i = f_i \quad i = 1, 2, \cdots, n \tag{2-72}$$

其中

$$\xi_i = \frac{\phi_j^T C \phi_i}{2\omega_i} \tag{2-73}$$

称为阻尼比.

运用 Duhamel 积分，式(2-72)的解为

$$\eta_i(t) = \eta_i(t_0) e^{-\xi_i \omega_i t} \cos \omega_d t + \frac{\dot{\eta}_i(t_0) + \xi_i \omega_i \eta_i(t_0)}{\omega_d} e^{-\xi_i \omega_i t} \sin \omega_d t + \frac{1}{\omega_d} \int_{t_0}^{t} e^{-\xi_i \omega_i (t-\tau)} \sin \omega_d (t-\tau) f_i(\tau) d\tau \tag{2-74}$$

式中，$\omega_d = \omega_i \sqrt{1-\xi^2}$ 是系统的有阻尼固有频率，ω_i 是系统的无阻尼固有频率.

对于脉冲激励 $f_i(t) = F_{0i}\delta(t)$，$t_0 = 0$，式(2-74)的解为

$$\eta_i(t) = \eta_i(0) e^{-\xi_i \omega_i t} \cos \omega_d t + \frac{\dot{\eta}_i(0) + \xi_i \omega_i \eta_i(0)}{\omega_d} e^{-\xi_i \omega_i t} \sin \omega_d t + \frac{1}{\omega_d} F_{0i} e^{-\xi_i \omega_i t} \sin \omega_d t$$

对于阶跃激励 $f_i(t) = F_{0i}$，$t_0 = 0$，式(2-74)的解为

$$\eta_i(t) = \eta_i(0) e^{-\xi_i \omega_i t} \cos \omega_d t + \frac{\dot{\eta}_i(0) + \xi_i \omega_i \eta_i(0)}{\omega_d} e^{-\xi_i \omega_i t} \sin \omega_d t + \frac{F_{0i}}{\omega_d^2 + \xi_i^2 \omega_i^2} \left(1 - e^{-\xi_i \omega_i t} \cos \omega_d t - \frac{\xi_i \omega_i}{\omega_d} e^{-\xi_i \omega_i t} \sin \omega_d t\right)$$

对于简谐激励 $f_i(t) = F_{0i} \cos \omega (t - t_0)$，$t_0 = 0$，式(2-74)的解为

$$\eta_i(t) = \eta_i(0) e^{-\xi_i \omega_i t} \cos \omega_d t + \frac{\dot{\eta}_i(0) + \xi_i \omega_i \eta_i(0)}{\omega_d} e^{-\xi_i \omega_i t} \sin \omega_d t - X e^{-\xi_i \omega_i t} \left(\frac{\xi_i \omega_i \cos \psi + \omega \sin \psi}{\omega_d} \sin \omega_d t + \cos \psi \cos \omega_d t\right) + X \cos(\omega t - \psi)$$

其中

$$X=\frac{F_{0i}}{\omega_i^2}\frac{1}{\sqrt{(1-\gamma^2)^2+(2\xi_i\gamma)^2}}, \quad \psi=\arctan\frac{2\xi_i\gamma}{1-\gamma^2}$$

初始条件仍为

$$\boldsymbol{\eta}(0)=\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{\delta}(0), \quad \dot{\boldsymbol{\eta}}(0)=\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{M}\dot{\boldsymbol{\delta}}(0)$$

系统在实际坐标系下的响应

$$\boldsymbol{\delta}(t)=\boldsymbol{\Phi}\boldsymbol{\eta}(t)$$

或

$$\delta_i(t)=\sum_{j=1}^{n}\phi_{ij}\eta_j(t) \qquad (i=1,2,\cdots,n)$$

2.3 频响函数分析

2.3.1 比例阻尼系统(实模态分析)

对式(2-13)两边进行拉氏变换得

$$\left(\boldsymbol{M}s^2+\boldsymbol{C}s+\boldsymbol{K}\right)\overline{\boldsymbol{\delta}}(s)=\overline{\boldsymbol{F}}(s) \tag{2-75}$$

式中，$\overline{\boldsymbol{\delta}}(s)$和$\overline{\boldsymbol{F}}(s)$分别为$\boldsymbol{\delta}(t)$和$\boldsymbol{F}(t)$的拉氏变换.

由式(2-75)，得

$$\overline{\boldsymbol{\delta}}(s)=\boldsymbol{H}_d(s)\overline{\boldsymbol{F}}(s) \tag{2-76}$$

式中，$\boldsymbol{H}_d(s)$为结构的位移传递函数矩阵，其表达式为

$$\boldsymbol{H}_d(s)=\left(\boldsymbol{M}s^2+\boldsymbol{C}s+\boldsymbol{K}\right)^{-1} \tag{2-77}$$

由于振型的正交性，有

$$\begin{cases}\boldsymbol{I}=\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{\Phi}\\ \mathrm{diag}\left(2\xi_i\omega_i\right)=\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{\Phi}\\ \mathrm{diag}\left(\omega_i^2\right)=\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\Phi}\end{cases} \tag{2-78}$$

式中，diag表示对角矩阵.

由式(2-78)得

$$\begin{cases}\boldsymbol{M}=\left(\boldsymbol{\Phi}^{\mathrm{T}}\right)^{-1}\boldsymbol{\Phi}^{-1}\\ \boldsymbol{C}=\left(\boldsymbol{\Phi}^{\mathrm{T}}\right)^{-1}\mathrm{diag}\left(2\xi_i\omega_i\right)\boldsymbol{\Phi}^{-1}\\ \boldsymbol{K}=\left(\boldsymbol{\Phi}^{\mathrm{T}}\right)^{-1}\mathrm{diag}\left(\omega_i^2\right)\boldsymbol{\Phi}^{-1}\end{cases} \tag{2-79}$$

将式(2-79)代入式(2-78)得

$$\begin{aligned} \boldsymbol{H}_d(s) &= \left[\left(\boldsymbol{\Phi}^{\mathrm{T}}\right)^{-1}\operatorname{diag}\left(s^2+2\xi_i\omega_i s+\omega_i^2\right)\boldsymbol{\Phi}^{-1}\right]^{-1} \\ &= \boldsymbol{\Phi}\operatorname{diag}\left(1/\left(s^2+2\xi_i\omega_i s+\omega_i^2\right)\right)\boldsymbol{\Phi}^{\mathrm{T}} \\ &= \sum_{i=1}^{n}\frac{\boldsymbol{\phi}_i\boldsymbol{\phi}_i^{\mathrm{T}}}{s^2+2\xi_i\omega_i s+\omega_i^2} \end{aligned} \tag{2-80}$$

同样，可由拉氏变换性质得结构初态为静止时的结构速度和加速度的传递函数矩阵分别为

$$\boldsymbol{H}_v(s)=\sum_{i=1}^{n}\frac{s\boldsymbol{\phi}_i\boldsymbol{\phi}_i^{\mathrm{T}}}{s^2+2\xi_i\omega_i s+\omega_i^2} \tag{2-81}$$

$$\boldsymbol{H}_a(s)=\sum_{i=1}^{n}\frac{s^2\boldsymbol{\phi}_i\boldsymbol{\phi}_i^{\mathrm{T}}}{s^2+2\xi_i\omega_i s+\omega_i^2} \tag{2-82}$$

令 $s=\mathrm{j}\omega\left(\mathrm{j}=\sqrt{-1}\right)$，可得结构位移、速度和加速度的频响函数矩阵分别为

$$\boldsymbol{H}_d(\omega)=\boldsymbol{H}_d(s)\big|_{s=\mathrm{j}\omega}=\sum_{i=1}^{n}\frac{\boldsymbol{\phi}_i\boldsymbol{\phi}_i^{\mathrm{T}}}{\omega_i^2-\omega^2+\mathrm{j}2\xi_i\omega_i\omega} \tag{2-83}$$

$$\boldsymbol{H}_v(\omega)=\boldsymbol{H}_v(s)\big|_{s=\mathrm{j}\omega}=\sum_{i=1}^{n}\frac{\mathrm{j}\omega\boldsymbol{\phi}_i\boldsymbol{\phi}_i^{\mathrm{T}}}{\omega_i^2-\omega^2+\mathrm{j}2\xi_i\omega_i\omega} \tag{2-84}$$

$$\boldsymbol{H}_a(\omega)=\boldsymbol{H}_a(s)\big|_{s=\mathrm{j}\omega}=\sum_{i=1}^{n}\frac{-\omega^2\boldsymbol{\phi}_i\boldsymbol{\phi}_i^{\mathrm{T}}}{\omega_i^2-\omega^2+\mathrm{j}2\xi_i\omega_i\omega} \tag{2-85}$$

位移频响函数矩阵中的任一元素 $H_d^{pq}(\omega)$ 表示 q 点激励、 p 点位移响应的频响函数，其表达式为

$$H_d^{pq}(\omega)=\sum_{i=1}^{n}\frac{\phi_{pi}\phi_{qi}}{\omega_i^2-\omega^2+\mathrm{j}2\xi_i\omega_i\omega} \tag{2-86}$$

位移频响函数 $H_d^{pq}(\omega)$ 是复函数，可用实部和虚部表示为

$$H_d^{pq}(\omega)=\sum_{i=1}^{n}\phi_{pi}\phi_{qi}\left[\frac{\omega_i^2-\omega^2}{\left(\omega_i^2-\omega^2\right)^2+\left(2\xi_i\omega_i\omega\right)^2}+\mathrm{j}\frac{-2\xi_i\omega_i\omega}{\left(\omega_i^2-\omega^2\right)^2+\left(2\xi_i\omega_i\omega\right)^2}\right] \tag{2-87}$$

2.3.2　一般阻尼系统(复模态理论)

一般情况下，存在阻尼力与速度成正比的粘性阻尼，假定阻尼矩阵仍然是对称阵，但不满足比例粘性阻尼. 在构造空间中描述的运动微分方程移到所谓状态空间来描述的想法成为寻求状态空间中构造相应的模态空间，使方程找到解耦的一条途径. 对于式(2-13)，若阻尼矩阵不能对角化，为求解这一方程，可引入下列辅助方程：

$$\boldsymbol{M}\dot{\boldsymbol{\delta}}-\boldsymbol{M}\dot{\boldsymbol{\delta}}=\boldsymbol{0} \tag{2-88}$$

将式(2-88)与式(2-13)联立，并以矩阵形式表示，可得

$$\begin{bmatrix}\boldsymbol{C} & \boldsymbol{M}\\ \boldsymbol{M} & \boldsymbol{0}\end{bmatrix}\begin{Bmatrix}\dot{\boldsymbol{\delta}}\\ \ddot{\boldsymbol{\delta}}\end{Bmatrix}+\begin{bmatrix}\boldsymbol{K} & \boldsymbol{0}\\ \boldsymbol{0} & -\boldsymbol{M}\end{bmatrix}\begin{Bmatrix}\boldsymbol{\delta}\\ \dot{\boldsymbol{\delta}}\end{Bmatrix}=\begin{Bmatrix}\boldsymbol{F}\\ \boldsymbol{0}\end{Bmatrix} \tag{2-89}$$

令 $\tilde{\delta}=\begin{Bmatrix}\delta \\ \dot{\delta}\end{Bmatrix}_{(2n\times1)}$ 称为状态向量.

式(2-89)可改写为

$$A\dot{\tilde{\delta}}+B\tilde{\delta}=\tilde{F} \tag{2-90}$$

式(2-90)称为状态方程，其中

$$A=\begin{bmatrix}C & M \\ M & 0\end{bmatrix},\quad B=\begin{bmatrix}K & 0 \\ 0 & -M\end{bmatrix},\quad \tilde{F}=\begin{Bmatrix}F \\ 0\end{Bmatrix} \tag{2-91}$$

考察系统式(2-90)的自由运动时，有

$$A\dot{\tilde{\delta}}+B\tilde{\delta}=0 \tag{2-92}$$

设式(2-92)的解为

$$\tilde{\delta}=\Psi e^{\lambda t},\quad \dot{\tilde{\delta}}=\Psi\lambda e^{\lambda t} \tag{2-93}$$

将式(2-93)代入式(2-92)，得

$$(A\lambda+B)\begin{Bmatrix}\Psi \\ \Psi\lambda\end{Bmatrix}=0 \tag{2-94}$$

求解式(2-94)，可得 $2n$ 个复特征值和特征向量，它们分别记为

$$\lambda_1,\lambda_2,\cdots,\lambda_n,\lambda_1^*,\lambda_2^*,\cdots,\lambda_n^*$$

$$\begin{Bmatrix}\Psi_1 \\ \Psi_1\lambda_1\end{Bmatrix},\begin{Bmatrix}\Psi_2 \\ \Psi_2\lambda_2\end{Bmatrix},\cdots,\begin{Bmatrix}\Psi_n \\ \Psi_n\lambda_n\end{Bmatrix},\begin{Bmatrix}\Psi_1^* \\ \Psi_1^*\lambda_1^*\end{Bmatrix},\begin{Bmatrix}\Psi_2^* \\ \Psi_2^*\lambda_2^*\end{Bmatrix},\cdots,\begin{Bmatrix}\Psi_n^* \\ \Psi_n^*\lambda_n^*\end{Bmatrix}$$

对第 r 阶模态，有

$$\lambda_r,\quad \lambda_r^*,\quad \begin{Bmatrix}\Psi_r \\ \Psi_r\lambda_r\end{Bmatrix},\quad \begin{Bmatrix}\Psi_r^* \\ \Psi_r^*\lambda_r^*\end{Bmatrix}$$

令

$$\begin{Bmatrix}\Psi_r \\ \Psi_r\lambda_r\end{Bmatrix}=\bar{\Psi}_r,\quad \begin{Bmatrix}\Psi_r^* \\ \Psi_r^*\lambda_r^*\end{Bmatrix}=\bar{\Psi}_r^*$$

不难证明，式(2-92)的特征向量 $\bar{\Psi}_r$ 具有关于 A 与 B 的加权正交性，即

$$\begin{cases}\bar{\Psi}_r^{\mathrm{T}}A\bar{\Psi}_s=\begin{cases}a_r & (r=s) \\ 0 & (r\neq s)\end{cases} \\ \bar{\Psi}_r^{\mathrm{T}}B\bar{\Psi}_s=\begin{cases}b_r & (r=s) \\ 0 & (r\neq s)\end{cases}\end{cases} \tag{2-95}$$

对全部模态而言，模态矩阵为

$$\bar{\Psi}=\begin{bmatrix}\bar{\Psi}_1 & \bar{\Psi}_2 & \cdots & \bar{\Psi}_n\end{bmatrix}$$

$$\bar{\Psi}^*=\begin{bmatrix}\bar{\Psi}_1^* & \bar{\Psi}_2^* & \cdots & \bar{\Psi}_n^*\end{bmatrix}$$

特征矩阵为

$$\begin{bmatrix} \Lambda & 0 \\ 0 & \Lambda^* \end{bmatrix} = \begin{bmatrix} \ddots & & & & \\ & \lambda_r & & & \\ & & \ddots & & \\ & & & \lambda_r^* & \\ & & & & \ddots \end{bmatrix} \quad (r=1,2,\cdots,n)$$

令

$$\boldsymbol{\Phi} = \begin{bmatrix} \bar{\boldsymbol{\Psi}} & \bar{\boldsymbol{\Psi}}^* \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Psi} & \boldsymbol{\Psi}^* \\ \boldsymbol{\Psi}\boldsymbol{\Lambda} & \boldsymbol{\Psi}^*\boldsymbol{\Lambda}^* \end{bmatrix}$$

则对于全部模态而言，正交性条件可由下式表示

$$\begin{cases} \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{A} \boldsymbol{\Phi} = \operatorname{diag}\left[a_1 \cdots a_n \ a_1^* \cdots a_n^* \right] \\ \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{B} \boldsymbol{\Phi} = \operatorname{diag}\left[b_1 \cdots b_n \ b_1^* \cdots b_n^* \right] \end{cases} \tag{2-96}$$

对状态向量 $\tilde{\boldsymbol{\delta}}$ 用复模态坐标进行变换，可得

$$\tilde{\boldsymbol{\delta}} = \begin{Bmatrix} \boldsymbol{\delta} \\ \dot{\boldsymbol{\delta}} \end{Bmatrix} = \begin{bmatrix} \boldsymbol{\Psi} & \boldsymbol{\Psi}^* \\ \boldsymbol{\Psi}\boldsymbol{\Lambda} & \boldsymbol{\Psi}^*\boldsymbol{\Lambda}^* \end{bmatrix} \begin{Bmatrix} \boldsymbol{Q} \\ \boldsymbol{Q}^* \end{Bmatrix} \tag{2-97}$$

式中，$\boldsymbol{Q}$，$\boldsymbol{Q}^*$ 为复模态坐标.

将式(2-97)代入式(2-90)，并左乘 $\begin{bmatrix} \boldsymbol{\Psi} & \boldsymbol{\Psi}^* \\ \boldsymbol{\Psi}\boldsymbol{\Lambda} & \boldsymbol{\Psi}^*\boldsymbol{\Lambda}^* \end{bmatrix}^{\mathrm{T}}$，并由式(2-96)的正交性条件可得

$$\begin{bmatrix} a_1 & & & & & \\ & \ddots & & & & \\ & & a_n & & & \\ & & & a_1^* & & \\ & & & & \ddots & \\ & & & & & a_n^* \end{bmatrix} \begin{Bmatrix} \dot{\boldsymbol{Q}} \\ \dot{\boldsymbol{Q}}^* \end{Bmatrix} + \begin{bmatrix} b_1 & & & & & \\ & \ddots & & & & \\ & & b_n & & & \\ & & & b_1^* & & \\ & & & & \ddots & \\ & & & & & b_n^* \end{bmatrix} \begin{Bmatrix} \boldsymbol{Q} \\ \boldsymbol{Q}^* \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{\Psi}^{\mathrm{T}} \boldsymbol{F} \\ \boldsymbol{\Psi}^{*\mathrm{T}} \boldsymbol{F} \end{Bmatrix} \tag{2-98}$$

对于第 r 阶模态，可写成

$$\begin{cases} a_r \dot{q}_r + b_r q_r = \boldsymbol{\Psi}_r^{\mathrm{T}} \boldsymbol{F} \\ a_r^* \dot{q}_r^* + b_r^* q_r^* = \boldsymbol{\Psi}_r^{*\mathrm{T}} \boldsymbol{F} \end{cases} \tag{2-99}$$

式(2-99)的解可由 Duhamel 积分求得

$$q_r = q_{r0} \mathrm{e}^{\lambda_r t} + \frac{1}{a_r} \int_0^t \mathrm{e}^{\lambda_r (t-\tau)} \boldsymbol{\Psi}_r^{\mathrm{T}} \boldsymbol{F}(\tau) \mathrm{d}\tau \tag{2-100}$$

$$q_r^* = q_{r0}^* \mathrm{e}^{\lambda_r^* t} + \frac{1}{a_r^*} \int_0^t \mathrm{e}^{\lambda_r^* (t-\tau)} \boldsymbol{\Psi}_r^{*\mathrm{T}} \boldsymbol{F}(\tau) \mathrm{d}\tau \tag{2-101}$$

设初始状态 $(t=0)$ 为振动平衡位置，即 $q_{r0} = \dot{q}_{r0} = 0$，则系统受迫振动的响应向量可写成

$$\begin{aligned} \boldsymbol{\delta} &= \boldsymbol{\Psi}\boldsymbol{Q} + \boldsymbol{\Psi}^*\boldsymbol{Q}^* = \sum_{r=1}^{n} \boldsymbol{\Psi}_r q_r + \sum_{r=1}^{n} \boldsymbol{\Psi}_r^* q_r^* \\ &= \sum_{r=1}^{n} \left[\frac{\boldsymbol{\Psi}_r \boldsymbol{\Psi}_r^{\mathrm{T}}}{a_r} \int_0^t \mathrm{e}^{\lambda_r (t-\tau)} \boldsymbol{\Psi}_r^{\mathrm{T}} \boldsymbol{F}(\tau) \mathrm{d}\tau + \frac{\boldsymbol{\Psi}_r^* \boldsymbol{\Psi}_r^{*\mathrm{T}}}{a_r^*} \int_0^t \mathrm{e}^{\lambda_r^* (t-\tau)} \boldsymbol{\Psi}_r^{*\mathrm{T}} \boldsymbol{F}(\tau) \mathrm{d}\tau \right] \end{aligned} \tag{2-102}$$

对式(2-102)两边做拉氏变换，得

$$\delta(s)=\sum_{r=1}^{n}\left(\frac{\Psi_r\Psi_r^{\mathrm{T}}}{a_r}\frac{1}{s-\lambda_r}+\frac{\Psi_r^*\Psi_r^{*\mathrm{T}}}{a_r^*}\frac{1}{s-\lambda_r^*}\right)F(s) \tag{2-103}$$

因此，传递函数矩阵为

$$H(s)=\sum_{r=1}^{n}\left(\frac{\Psi_r\Psi_r^{\mathrm{T}}}{a_r}\frac{1}{s-\lambda_r}+\frac{\Psi_r^*\Psi_r^{*\mathrm{T}}}{a_r^*}\frac{1}{s-\lambda_r^*}\right) \tag{2-104}$$

则系统的频响函数矩阵为

$$H(\omega)=\sum_{r=1}^{n}\left(\frac{\Psi_r\Psi_r^{\mathrm{T}}}{a_r(\mathrm{j}\omega-\lambda_r)}+\frac{\Psi_r^*\Psi_r^{*\mathrm{T}}}{a_r^*(\mathrm{j}\omega-\lambda_r^*)}\right) \tag{2-105}$$

其中，q 点激励、p 点位移响应的复模态频响函数可表达为

$$H_{pq}(\omega)=\sum_{r=1}^{n}\left(\frac{\Psi_{pr}\Psi_{qr}}{a_r(\mathrm{j}\omega-\lambda_r)}+\frac{\Psi_{pr}^*\Psi_{qr}^*}{a_r^*(\mathrm{j}\omega-\lambda_r^*)}\right) \tag{2-106}$$

2.4 应用问题与 MATLAB 程序

2.4.1 结构动力特性分析

【例 2.1】 平面 61 杆桁架，如图 2.1 所示．

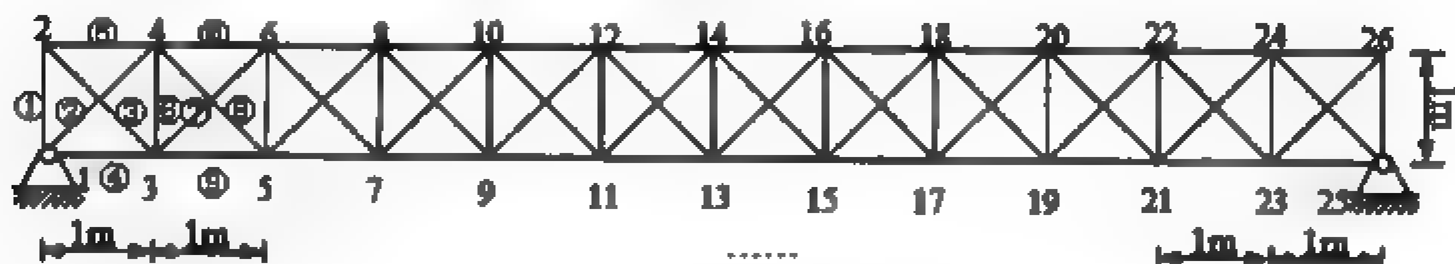


图 2.1 平面 61 杆桁架

平面 61 杆桁架的材料特性为：弹性模量 $E=2.1\times10^{11}\,\mathrm{Pa}$，质量密度 $\rho=7\,300\,\mathrm{kg/m^3}$，所有单元的横截面积为 $1\times10^{-4}\,\mathrm{m^2}$．分别采用 3 种方法计算结构的前 9 阶频率：矢量迭代法、子空间迭代法和 MATLAB 提供的函数 eigs．用 MATLAB 编程计算，其中主要用到的 MATLAB 函数(见 2.4.4 节)有：

- Inviter(k,m,p,epsilon)——用矢量迭代法计算结构的特征值和特征向量．
- Ssiter(k,m,p,epsilon)——用子空间迭代法计算结构的特征值和特征向量．

M 文件如下：

```
%-----------------------------------------------------------------------
% Ex.2.1 Plane 61-bar truss
%-----------------------------------------------------------------------
% To compute the first nine eigenvalues of plane 61-bar truss
%
E=2.1e11;
A=1e-4;
```

```
density=7.3e3;
node_number=26;
element_number=61;
nc=zeros(26,2);
nc(1,1)=0;nc(1,2)=0;
nc(2,1)=0;nc(2,2)=1;
for ig=1:12
    for jg=1:2
        nc(ig*2+jg,1)=nc(jg,1)+ig;           %node coordinate
        nc(ig*2+jg,2)=nc(jg,2);
    end
end
en=zeros(61,2);
en(1,1)=1;    en(1,2)=2;                     %en:element_node
en(2,1)=1;    en(2,2)=4;
en(3,1)=2;    en(3,2)=3;
en(4,1)=1;    en(4,2)=3;
en(5,1)=2;    en(5,2)=4;
for i=1:11
   for j=1:5
      en(i*5+j,1)=en(j,1)+2*i;
      en(i*5+j,2)=en(j,2)+2*i;
   end
end
en(61,1)=25;    en(61,2)=26;
ed(1:node_number,1:2)=1;                     %ed:element_displacement
constraint=[1,1;1,2;25,2];
for loopi=1:length(constraint);
    ed(constraint(loopi,1),constraint(loopi,2))=0;
end
dof=0;
for loopi=1:node_number
     for loopj=1:2
        if ed(loopi,loopj)~=0
           dof=dof+1;
           ed(loopi,loopj)=dof;
        end
     end
end
ek=E*A*[1 0 -1 0;0 0 0 0;-1 0 1 0;0 0 0 0];    %element_stiffness_matrix
em=(density*A)/6*[2 0 1 0; ...               %element_mass_matrix
                 0 2 0 1; ...
                 1 0 2 0; ...
                 0 1 0 2];
k(1:dof,1:dof)=0;                            %structural_stiffness_matrix
m=k;                                         %structural_mass_matrix,same size
with k
```

```
theta(1:61)=0;
el(1:61)=0;
e2s(1:4)=0;                    % index of transform the element
displament number to structure
for loopi=1:element_number
    for zi=1:2
        e2s((zi-1)*2+1)=ed(en(loopi,zi),1);
        e2s((zi-1)*2+2)=ed(en(loopi,zi),2);
    end
el(loopi)=sqrt((nc(en(loopi,1),1)-nc(en(loopi,2),1))^2
+(nc(en(loopi,1),2)-nc(en(loopi,2),2))^2);
    theta(loopi)=asin((nc(en(loopi,1),2)-nc(en(loopi,2),2))/el(loopi));
    lmd=[cos(theta(loopi)) sin(theta(loopi)); -sin(theta(loopi))
    cos(theta(loopi))];
    t=[lmd zeros(2); zeros(2) lmd];
    dk=t'*ek*t/el(loopi);
    dm=t'*em*t*el(loopi);
    for jx=1:4
        for jy=1:4
            if(e2s(jx)*e2s(jy)~=0)
                k(e2s(jx),e2s(jy))=k(e2s(jx),e2s(jy))+dk(jx,jy);
                m(e2s(jx),e2s(jy))=m(e2s(jx),e2s(jy))+dm(jx,jy);
            end
        end
    end
end
% Inverse iterative method
p=9;
epsilon=1e-5;
[v,d]=Inviter(k,m,p,epsilon);     %d:eigenvalue  v:eigenvector
frequency=sqrt(d)/(2*pi);
% Suspace iterative method
p=9;
epsilon=1e-5;
[v,d]=Ssiter(k,m,p,epsilon);
frequency=sqrt(d)/(2*pi);
% MATLAB function eigs
 [v,d] =eigs(k,m,9,'SM');
frequency=sqrt(diag(d))/(2*pi);
[frequency,indexf]=sort(frequency)
v=v(:,indexf);
Factor=diag(v'*m*v);
v=v*inv(sqrt(diag(Factor)));      % normalize eigenvector
```

3 种方法运行结果比较如下：

```
frequency =
    method 1              method 2            method 3
```

16.4815	16.4815	16.4815
54.9565	54.9564	54.9564
73.7466	73.7467	73.7467
132.1518	132.1518	132.1518
193.0646	193.0635	193.0635
222.2496	222.2514	222.2514
302.8284	302.8278	302.8278
337.6138	337.6155	337.6155
404.0057	404.0046	404.0042

2.4.2　结构时域动力响应分析

【例 2.2】　平面九杆桁架，如图 2.2 所示.

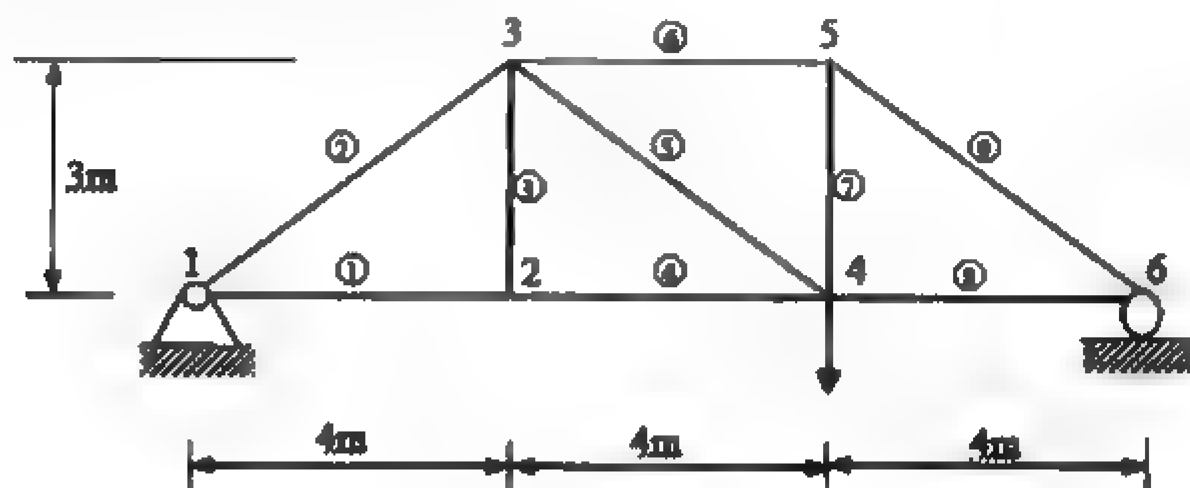


图 2.2　平面九杆桁架

平面九杆桁架的材料特性为：弹性模量 $E = 2.0\times10^{11}$Pa，质量密度 $\rho = 7\,860\text{kg/m}^3$，所有单元的横截面积为 $2.5\times10^{-3}\text{m}^2$．在节点 4 竖直向上加有一大小为 200N 的阶跃力．采用 6 种方法计算其响应，节点 4 竖直方向的位移分别如图 2.3(a)~(f)所示，该 6 种方法依次为：中心差分法、Houbolt 法、Wilson-θ 法、Newmark 法、状态空间法、振型叠加法．用 MATLAB 编程计算，其中主要用到的 MATLAB 函数(见 2.4.4 节)有：

- TransResp1(kk1,cc1,mm1,ft0,bcdof1,nt,dt,q0,dq0)——中心差分法计算响应.
- TransResp3(kk1,cc1,mm1,ft0,bcdof1,nt,dt,q0,dq0)——Houbolt 法计算响应.
- TransResp4(kk1,cc1,mm1,ft0,bcdof1,nt,dt,q0,dq0)——Wilson-θ 法计算响应.
- TransResp5(kk1,cc1,mm1,ft0,bcdof1,nt,dt,q0,dq0)——Newmark 法计算响应.
- Modalresponse(m,k,Fu,u,time,Cy,q0,dq0,nummode)——模态叠加法计算响应.

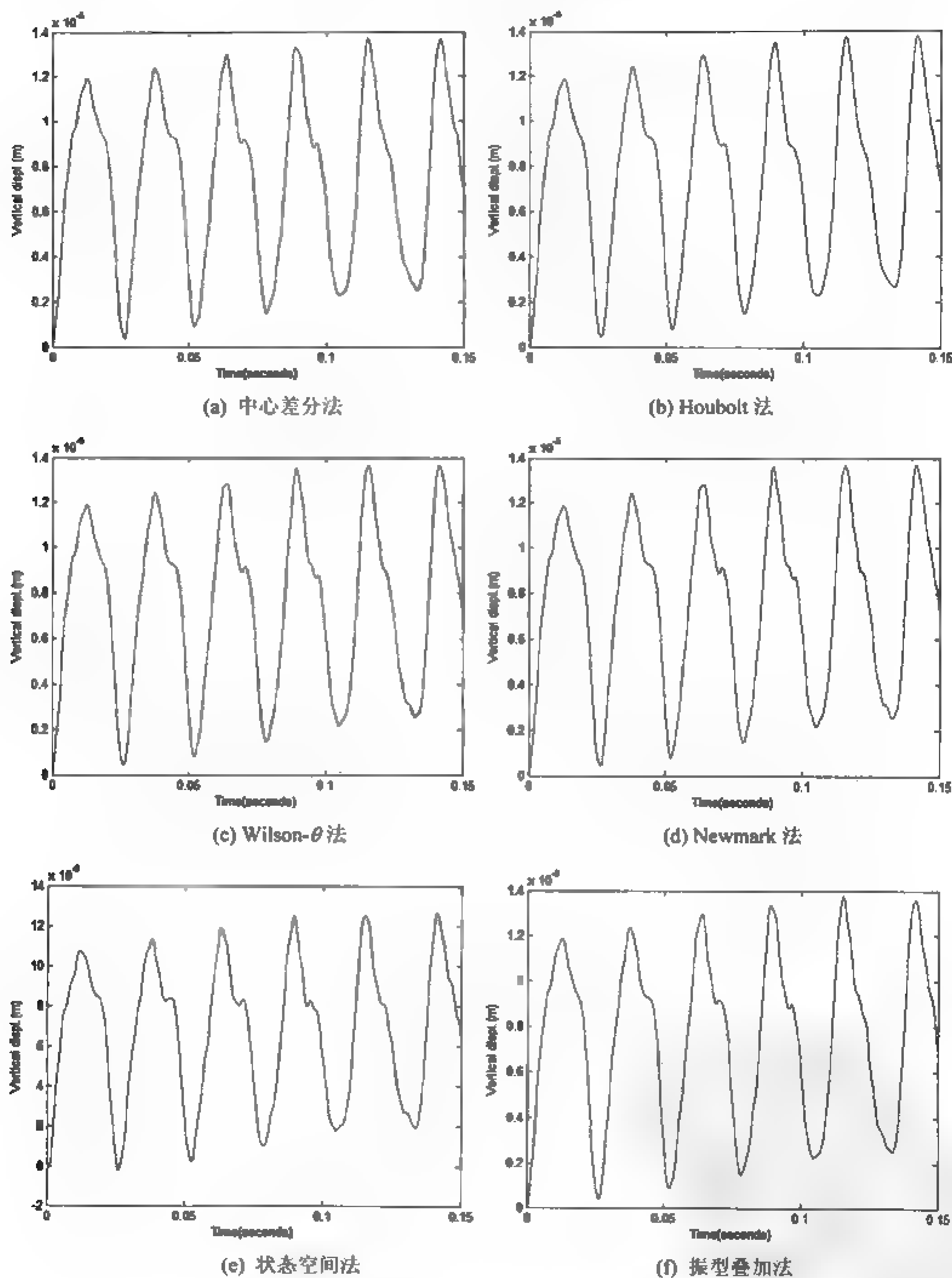
```
%-------------------------------------------------------------
% Ex.2.2 plane 9-bar truss
%-------------------------------------------------------------
% To compute the dynamic response of node 4 on the vertical direction
%
%
```

```
E=2.0e11;
A=2.5e-3;
density=7860;
node_number=6;
element_number=9;
nc=[0 0;4 0;4 3;8 0;8 3;12 0];          %node coordinate
en=[1 2;1 3;2 3;2 4;3 4;3 5;4 5;4 6;5 6];   % element_node
ed(1:node_number,1:2)=1;          % element_displacement
dof=0;
for loopi=1:node_number
     for loopj=1:2
           dof=dof+1;
           ed(loopi,loopj)=dof;
    end
end
ek=E*A*[1 0 -1 0;0 0 0 0;-1 0 1 0;0 0 0 0]; % element_stiffness_matrix
em=(density*A)/6*[2 0 1 0; ...          %element_mass_matrix
                 0 2 0 1; ...
                1 0 2 0; ...
                0 1 0 2];
k(1:dof,1:dof)=0;                    % structural_stiffness_matrix
m=k;                           % structural_mass_matrix, same size  with  k
theta(1:9)=0;
el(1:9)=0;
e2s(1:4)=0; %index of transform the element displament number to structural
for loopi=1:element_number
    for zi=1:2
        e2s((zi-1)*2+1)=ed(en(loopi,zi),1);
        e2s((zi-1)*2+2)=ed(en(loopi,zi),2);
    end
el(loopi)=sqrt((nc(en(loopi,1),1)-nc(en(loopi,2),1))^2
+(nc(en(loopi,1),2)-nc(en(loopi,2),2))^2);
    theta(loopi)=asin((nc(en(loopi,1),2)-nc(en(loopi,2),2))/el(loopi));
    lmd=[cos(theta(loopi)) sin(theta(loopi)); -sin(theta(loopi))
    cos(theta(loopi))];
    t=[lmd zeros(2); zeros(2) lmd];
    dk=t'*ek*t/el(loopi);
    dm=t'*em*t*el(loopi);
    for jx=1:4
        for jy=1:4
               k(e2s(jx),e2s(jy))=k(e2s(jx),e2s(jy))+dk(jx,jy);
               m(e2s(jx),e2s(jy))=m(e2s(jx),e2s(jy))+dm(jx,jy);
        end
    end
end
c=0*k+0*m;
nt=1500;dt=0.0001;
time=0:dt:nt*dt;
```

```
q0=zeros(dof,1);
dq0=zeros(dof,1);
bcdof=zeros(dof,1);
fd=zeros(dof,nt);
for i=1:nt
    fd(6,i)=200;
end
[acc,vel,dsp]=TransResp1(k,c,m,fd,bcdof,nt,dt,q0,dq0);
%[acc,vel,dsp]=TransResp3(k,c,m,fd,bcdof,nt,dt,q0,dq0);
%[acc,vel,dsp]=TransResp4(k,c,m,fd,bcdof,nt,dt,q0,dq0);
%[acc,vel,dsp]=TransResp5(k,c,m,fd,bcdof,nt,dt,q0,dq0);
plot(time, dsp(6,:))
 xlabel('Time(seconds)')
 ylabel(' Vertical displ. (m)')

% state-space method
 AG=[zeros(dof) eye(dof);-inv(m)*k zeros(dof)];
 BG=[zeros(dof,1);
     inv(m)*[0 0 0 0 0 1 0 0 0]'];
 CG=eye(2*dof);
 DG=0;
nt=1500;
det=0.0001;
time=0:det:(nt*det);
for i=1:(nt+1)
     u(i)=200;
end
G=ss(AG,BG,CG,DG);
dsp=lsim(G,u,time);
plot(time,dsp(:,6))
xlabel('Time(seconds)')
ylabel('Tip displ. (m)')

% modal analysis method
nt=1500;
det=0.0001;
time=0:det:(nt*det);
for i=1:(nt+1)
     u(i)=200;
end
Fu=[0 0 0 0 0 1 0 0 0]';
Cy=eye(dof);
q0=zeros(dof,1);
dq0=zeros(dof,1);
nummode=9;
[eta,dsp]=Modalresponse(m,k,Fu,u,time,Cy,q0,dq0,nummode);
plot(time,dsp(6,:))
xlabel('Time(seconds)')
ylabel(' Vertical displ. (m)')
```



(a) 中心差分法 (b) Houbolt 法

(c) Wilson-θ 法 (d) Newmark 法

(e) 状态空间法 (f) 振型叠加法

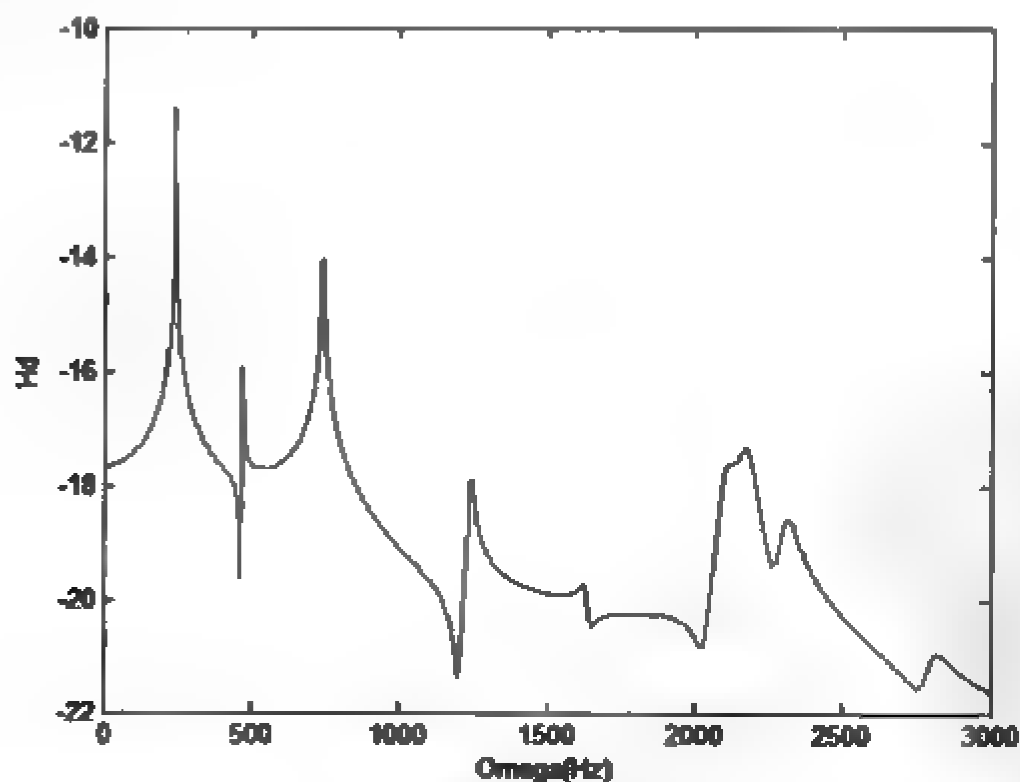
图 2.3 节点 4 竖直方向的位移

2.4.3 结构频响函数分析

【例 2.3】 平面九杆桁架。

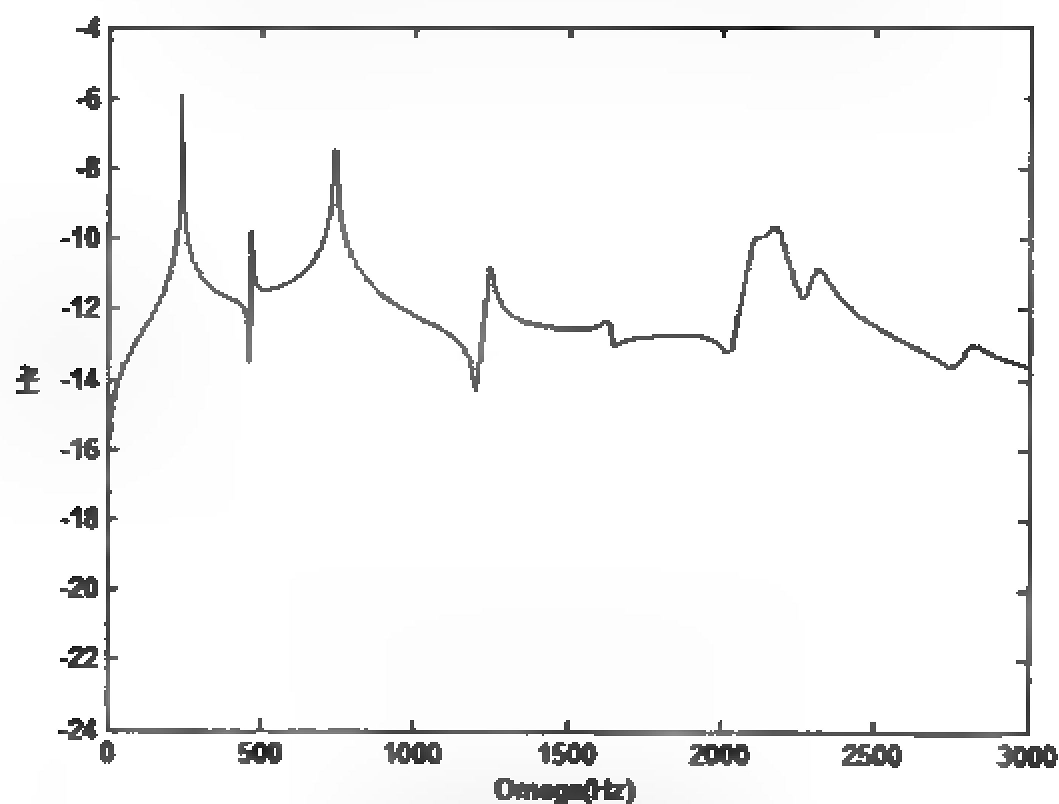
对例 2.2 的平面九杆桁架进行频响函数分析，假设阻尼为比例阻尼，即 $\boldsymbol{C}=1\times10^{-5}\boldsymbol{K}+1\times10^{-4}\boldsymbol{M}$，得到的节点 2 竖直方向与节点 5 竖直方向的位移、速度和加速度频响函数曲线分别如图 2.4(a)、(b)、(c)所示。

```
c=1e-5*k+1e-4*m;
nummode=9;
Omega=0:0.01:3000;
p=8;
q=2;
[Hd,Hv,Ha]=freqresponse(k,m,c,Omega,nummode,'all',p,q)
figure(1)
plot(Omega,log(abs(Hd)))
xlabel('Omega(Hz)')
ylabel('Hd')
figure(2)
plot(Omega,log(abs(Hv)))
xlabel('Omega(Hz)')
ylabel('Hv')
figure(3)
plot(Omega,log(abs(Ha)))
xlabel('Omega(Hz)')
ylabel('Ha')
```

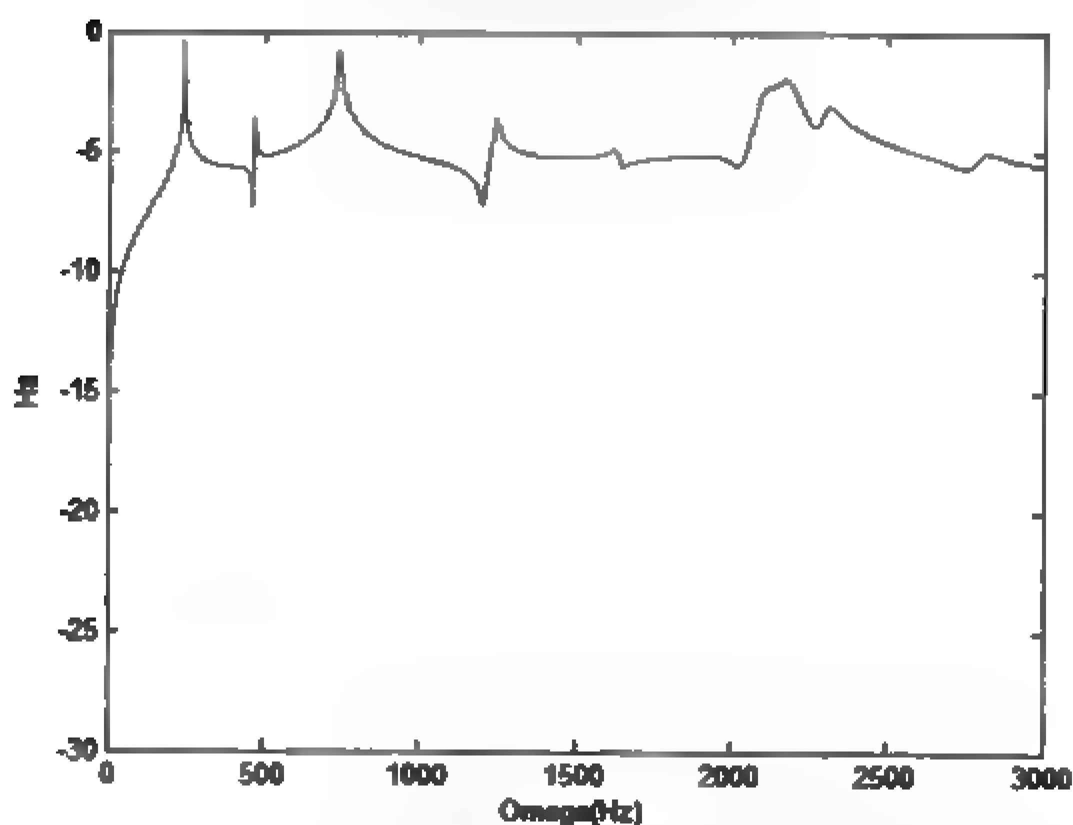


(a) 位移频响函数曲线

图 2.4　节点 2 竖直方向与节点 5 竖直方向的频响函数曲线



(b) 速度频响函数曲线



(c) 加速度频响函数曲线

图2.4 (续)

2.4.4 本实例所用的MATLAB函数

以下函数用矢量迭代法计算结构的前p阶特征值和特征向量.

```
function [v, d]=Inviter(K, M, p, epsilon)
%-----------------------------------------------------------------------
% Purpose:
%    The function calculates the first p eigenvalues and eigenvectors for
%    a structural system using inverse interation method.
```

```
% Synopsis:
%    [v, d]=Inviter(K, M, p, epsilon)
% Variable Description:
%    Input parameters
%    K - System siffness matrix
%    M - Sytem mass matrix
%    p - the number of calculated eigenvalues/eigenvectors
%    epsilon - the required precise
%    Output parameters
%    V - First p eigenvecotrs
%    d - First p eigenvalues
%---------------------------------------------------------------------
[n1,n2]=size(K);
KM=zeros(n1,1);
for i=1:n1
    KM(i)=K(i,i)/M(i,i);
end
[Y,I]=sort(KM,1,'ascend');
y=zeros(n1,p);
y(:,1)=diag(M);
for i=2:p
    y(I(i-1),i)=1;
end
d=zeros(p,1);
v=zeros(n1,p);

   for j=1:p
       flag=1;
    while flag
       if j>1
          for i=1:(j-1)
              xs=v(:,i)'*y(:,j)*v(:,i);
              xs=M*xs;
             y(:,j)=y(:,j)-xs;
          end
       end

       xiter=K\y(:,j);
       yiter=M*xiter;
       diter=transpose(xiter)*y(:,j)/(transpose(xiter)*yiter);
       if (abs(diter-d(j))/diter)<epsilon
          flag=0;
       else
          flag=1;
       end
         y(:,j)=yiter/((transpose(xiter)*yiter)^(1/2));
         d(j)=diter;
```

```
    end
      d(j)=diter;
      v(:,j)=xiter/((transpose(xiter)*yiter)^(1/2));
  end
%-----------------------------------------------------------------
%  The end
%-----------------------------------------------------------------
```

以下函数用子空间迭代法计算结构的前p阶特征值和特征向量.

```
function [v,d]=Ssiter(K,M,p,epsilon)
%----------------------------------------------------------------------
% Purpose:
%    The function calculates the first p eigenvalues and eigenvectors for
%    a structural system using subspace interation method.
% Synopsis:
%    [v,d]=Ssiter(K,M,p,epsilon)
% Variable Description:
%    Input parameters
%    K - System siffness matrix
%    M - Sytem mass matrix
%    p - the number of calculated eigenvalues/eigenvectors
%    epsilon - the required precise
%    Output parameters
%    V - First p eigenvecotrs
%    d - First p eigenvalues
%----------------------------------------------------------------------
% Initialize the first q eigenvectors
%---------------------------------
q=min(2*p,p+8);
[n1,n2]=size(K);
v0=zeros(n1,q);
MK=zeros(n1,1);
d0=zeros(p,1);
for i=1:n1
   MK(i)=M(i,i)/K(i,i);
end
[Y, I]=sort(MK,1,'descend');
v0(:,1)=1;
for i=2:q
   v0(I(i-1),i)=1/Y(i-1);
end
%-----------------------------------
iter=0;
flag=1;
while flag
   iter=iter+1;
   yiter=M*v0;
```

```
    xiter1=K\yiter;
    Km=(xiter1)'*K*xiter1;
    Mm=(xiter1)'*M*xiter1;
    [vm,dm]=eig(Km,Mm);    % compute eigenvalue and eigenvector in subspace
    [lambda,k]=sort(diag(dm));
    vm=vm(:,k);
    Factor=diag(vm'*Mm*vm);
    Vnorm=vm*inv(sqrt(diag(Factor)));        % normalize eigenvector
    v0=xiter1*Vnorm;
    diter=lambda(1:p);
    if abs(norm(diter)-norm(d0))/norm(diter)<epsilon  % termination condition
       flag=0;
    else
       flag=1;
    end
    d0=diter;
end
v=v0(:,1:p);
d=d0;
%--------------------------------------------------------------------------
% The end
%--------------------------------------------------------------------------
```

以下函数用中心差分法计算结构的动响应.

```
function [acc,vel,dsp]=TransResp1(kk,cc,mm,fd,bcdof,nt,dt,q0,dq0)
%--------------------------------------------------------------------------
% Purpose:
%    The function subroutine TransResp1.m calculates transient response of
%    a structural system using central difference scheme.
% Synopsis:
%    [acc,vel,dsp]=TransResp1(kk,cc,mm,fd,bcdof,nt,dt,q0,dq0)
% Variable Description:
%    Input parameters
%      kk, cc, mm - stiffness, damping and mass matrices
%      fd - Input or forcing influence matrix
%      bcdof - Boundary condition dofs vector
%      nt - Number of time steps
%      dt - Time step size
%      q0, dq0 - Initial condition vectors
%    Output parameters
%      acc - Acceleration response
%      vel - Velocity response
%      dsp - Displacement response
%--------------------------------------------------------------------------
% (1) initial condition
%--------------------------------------------------------------------------
[sdof,n2]=size(kk);
```

```
dsp=zeros(sdof,nt);                          % displacement matrix
vel=zeros(sdof,nt);                          % velocity matrix
acc=zeros(sdof,nt);                          % acceleration matrix

dsp(:,1)=q0;                                 % initial displacement
vel(:,1)=dq0;                                % initial velocity
% --------------------------------------------------------------------
% (2) central difference scheme for time integration
%---------------------------------------------------------------------
acc(:,1)=inv(mm)*(fd(:,1)-kk*dsp(:,1)-cc*vel(:,1));
                          % compute the initial acceleration (t=0)
dsp0=dsp(:,1)-vel(:,1)*dt+0.5*acc(:,1)*dt^2;
                          % compute the fictitious displacement at time -dt
ekk=mm/dt^2+cc/(2*dt);
                          % compute the effective stiffness matrix
%-----------------------------------------
% (2.1) first step of the central difference scheme
%-----------------------------------------
efd=fd(:,1)-(kk-2*mm/dt^2)*dsp(:,1)-(mm/dt^2-cc/(2*dt))*dsp0;
                          %  compute the effective force vector
dsp(:,1+1)=inv(ekk)*efd;
                          % find the dsp at 1+dt
for i=1:sdof    % assign zero to acc, vel, dsp of the dofs associated with bc
  if bcdof(i)==1
    dsp(i,1)=0;
    dsp(i,2)=0;
    vel(i,1)=0;
    acc(i,1)=0;
  end
end
%--------------------------------------------------
% (2.2) subsequent steps of the central difference scheme
%--------------------------------------------------
for it=2:nt                     % loop for each time step after first step
  efd=fd(:,it)-(kk-2*mm/dt^2)*dsp(:,it)-(mm/dt^2-cc/(2*dt))*dsp(:,it-1);
                                %  compute the effective force vector
  dsp(:,it+1)=inv(ekk)*efd;                 % find the dsp at t+dt
  acc(:,it)=(dsp(:,it+1)-2*dsp(:,it)+dsp(:,it-1))/dt^2;
                                % find the acc at t
  vel(:,it)=(dsp(:,it+1)-dsp(:,it-1))/(2*dt);
                                % find the vel at t

  for i=1:sdof  % assign zero to acc, vel, dsp of the dofs associated with bc
    if bcdof(i)==1
      dsp(i,it)=0;
      dsp(i,it+1)=0;
```

```
      vel(i,it)=0;
      acc(i,it)=0;
    end
  end

end

  acc(:,it+1)=acc(:,it); vel(:,it+1)=vel(:,it);

if cc(1,1)==0
  disp('The transient response results of undamping system')
else
  disp('The transient response results of damping system')
end

disp('The method is central difference scheme 1')
%--------------------------------------------------------------------
%     The end
%--------------------------------------------------------------------
```

以下函数用 Houbolt 法计算结构的动响应.

```
function [acc,vel,dsp]=TransResp3(kk,cc,mm,fd,bcdof,nt,dt,q0,dq0)
%--------------------------------------------------------------------
%  Purpose:
%     The function subroutine TransResp3.m calculates transient response of
%     a structural system using Houbolt integration scheme.
%  Synopsis:
%     [acc,vel,dsp]=TransResp3(kk,cc,mm,fd,bcdof,nt,dt,q0,dq0)
%  Variable Description:
%     Input parameters
%       kk, cc, mm - stiffness, damping and mass matrices
%       fd - Input or forcing influence matrix
%       bcdof - Boundary condition dofs vector
%       nt - Number of time steps
%       dt - Time step size
%       q0, dq0 - Initial condition vectors
%     Output parameters
%       acc - Acceleration response
%       vel - Velocity response
%       dsp - Displacement response
%--------------------------------------------------------------------
%  (1) initial condition
%--------------------------------------------------------------------
[sdof,n2]=size(kk);

dsp=zeros(sdof,nt);                             % displacement matrix
vel=zeros(sdof,nt);                             % velocity matrix
```

```
acc=zeros(sdof,nt);                              % acceleration matrix

dsp(:,1)=q0;                                     % initial displacement
vel(:,1)=dq0;                                    % initial velocity
%------------------------------------------------------------------------
%  (2) calculate the initial integration using central difference scheme
%------------------------------------------------------------------------
%----------------------------------------
%  (2.1) calculate the displacement at time -dt
%----------------------------------------
acc(:,1)=inv(mm)*(fd(:,1)-kk*dsp(:,1)-cc*vel(:,1));
                        % compute the initial acceleration (t=0)
dsp0=dsp(:,1)-vel(:,1)*dt+0.5*acc(:,1)*dt^2;
                        % compute the fictitious displacement at time -dt

ekk=mm/dt^2+cc/(2*dt);         % compute the effective stiffness matrix
%----------------------------------------
%  (2.2) calculate the displacement at time t+dt
%----------------------------------------
it=1;

efd=fd(:,it)-(kk-2*mm/dt^2)*dsp(:,it)-(mm/dt^2-cc/(2*dt))*dsp0;
                                       %  compute the effective force vector
dsp(:,it+1)=inv(ekk)*efd;                        % find the dsp at t+dt
               % assign zero to acc, vel, dsp of the dofs associated with bc
for i=1:sdof
  if bcdof(i)==1
    dsp(i,1)=0;
    dsp(i,2)=0;
    vel(i,1)=0;
    acc(i,1)=0;
  end
end
%-----------------------------------------
%  (2.3) calculate the displacement at time t+2dt
%-----------------------------------------
for it=2:3                          % loop for two steps after first step
  efd=fd(:,it)-(kk-2*mm/dt^2)*dsp(:,it)-(mm/dt^2-cc/(2*dt))*dsp(:,it-1);
                                    %  compute the effective force vector
  dsp(:,it+1)=inv(ekk)*efd;                      % find the dsp at t+dt

  acc(:,it)=(dsp(:,it+1)-2*dsp(:,it)+dsp(:,it-1))/dt^2;  % find the acc at t
  vel(:,it)=(dsp(:,it+1)-dsp(:,it-1))/(2*dt);      % find the vel at t

  for i=1:sdof   % assigh zero acc, vel, dsp of the dofs associated with bc
    if bcdof(i)==1
      dsp(i,it)=0;
```

```
      dsp(i,it+1)=0;
      vel(i,it)=0;
      acc(i,it)=0;
    end
  end

end
%------------------------------------------------------------------------
%  (3) subsequent steps of the central difference scheme
%------------------------------------------------------------------------
ekk=kk+cc*11/(6*dt)+mm*2/(dt*dt);

for it=3:nt                    % loop for each time step after initial step
                               %  compute the effective force vector
  cfm=dsp(:,it)*5/dt^2-dsp(:,it-1)*4/dt^2+dsp(:,it-2)*1/dt^2;
  cfc=dsp(:,it)*3/dt-dsp(:,it-1)*3/(2*dt)+dsp(:,it-2)*1/(3*dt);

  efd=fd(:,it+1)+mm*cfm+cc*cfc;

  dsp(:,it+1)=inv(ekk)*efd;                        % find the dsp at t+dt
  acc(:,it)=(2*dsp(:,it+1)-5*dsp(:,it)+4*dsp(:,it-1)+dsp(:,it-2))/dt^2;
                                                   % find the acc at t+dt
  vel(:,it)=(11*dsp(:,it+1)-18*dsp(:,it)+9&dsp(:,it-1)-2*dsp(:,it-
2))/(6*dt);
                                                   % find the vel at t+dt
  for i=1:sdof  % assign zero to acc, vel, dsp of the dofs associated with bc
    if bcdof(i)==1
      dsp(i,it)=0;
      dsp(i,it+1)=0;
      vel(i,it)=0;
      acc(i,it)=0;
    end
  end

end

if cc(1,1)==0
  disp('The transient response results of undamping system')
else
  disp('The transient response results of damping system')
end
disp('The method is Houbolt integration scheme')
%------------------------------------------------------------------------
%    The end
% -----------------------------------------------------------------------
```

以下函数用 Wilson-θ 法计算结构的动响应.

```
function [acc,vel,dsp]=TransResp4(kk,cc,mm,fd,bcdof,nt,dt,q0,dq0)
%--------------------------------------------------------------------------
%  Purpose:
%     The function subroutine TransResp4.m calculates transient response of
%     a structural system using Wilson θ integration scheme.
%  Synopsis:
%     [acc, vel, dsp]=TransResp4(kk,cc,mm,fd,bcdof,nt,dt,q0,dq0)
%  Variable Description:
%     Input parameters
%       kk, cc, mm - stiffness, damping and mass matrices
%       fd - Input or forcing influence matrix
%       bcdof - Boundary condition dofs vector
%       nt - Number of time steps
%       dt - Time step size
%       q0, dq0 - Initial condition vectors
%     Output parameters
%       acc - Acceleration response
%       vel - Velocity response
%       dsp - Displacement response
%--------------------------------------------------------------------------
%  (1) initial condition
%--------------------------------------------------------------------------
[sdof,n2]=size(kk);

dsp=zeros(sdof,nt);                                  % displacement matrix
vel=zeros(sdof,nt);                                  % velocity matrix
acc=zeros(sdof,nt);                                  % acceleration matrix

dsp(:,1)=q0;                                          % initial displacement
vel(:,1)=dq0;                                         % initial velocity

theta=1.4;                                          % select the parameters
%--------------------------------------------------------------------------
%  (2) Wilson θ integration scheme for time integration
%--------------------------------------------------------------------------
acc(:,1)=inv(mm)*(fd(:,1)-kk*dsp(:,1)-cc*vel(:,1));
                              % compute the initial acceleration (t=0)
ekk=kk+mm*(6/(theta*dt)^2)+cc*(3/(theta*dt));
                              % compute the effective stiffness matrix
for i=1:sdof   % assign zero to dsp, vel, acc of the dofs associated with bc
  if bcdof(i)==1
    dsp(i,1)=0;
    vel(i,1)=0;
    acc(i,1)=0;
  end
end
```

```
for it=1:nt                          % loop for each time step
  cfm=dsp(:,it)*(6/(theta*dt)^2)+vel(:,it)*(6/(theta*dt))+2*acc(:,it);
  cfc=dsp(:,it)*(3/(theta*dt))+2*vel(:,it)+acc(:,it)*(theta*dt/2);
  efd=fd(:,it)+theta*(fd(:,it+1)-fd(:,it))+mm*cfm+cc*cfc;
                                     %  compute the effective force vector
  dtheta=inv(ekk)*efd;               % find the displacement at time t+θdt

  acc(:,it+1)=(dtheta-dsp(:,it))*(6/(theta^3*dt^2))...
            -vel(:,it)*(6/(theta^2*dt))+acc(:,it)*(1-3/theta);
                                     % find the acceleration at time t+dt
  vel(:,it+1)=vel(:,it)+acc(:,it+1)*dt/2+acc(:,it)*dt/2;
                                     % find the velocity at time t+dt
  dsp(:,it+1)=dsp(:,it)+vel(:,it)*dt...
            +(acc(:,it+1)+2*acc(:,it))*(dt^2/6);
                                     % find the displacement at time t+dt

  for i=1:sdof   % assign zero to acc, vel, dsp of the dofs associated with bc
    if bcdof(i)==1
      dsp(i,it+1)=0;
      vel(i,it+1)=0;
      acc(i,it+1)=0;
    end
  end

end

if cc(1,1)==0
  disp('The transient response results of undamping system')
else
  disp('The transient response results of damping system')
end

disp('The method is Wilson θ integration')
%---------------------------------------------------------------------
%    The end
%---------------------------------------------------------------------
```

以下函数用 Newmark 法计算结构的动响应.

```
function [acc,vel,dsp]=TransResp5(kk,cc,mm,fd,bcdof,nt,dt,q0,dq0)
%---------------------------------------------------------------------
%  Purpose:
%    The function subroutine TransResp5.m calculates transient response of
%     a structural system using Newmark integration scheme.
%  Synopsis:
%     [acc,vel,dsp]=TransResp5(kk,cc,mm,fd,bcdof,nt,dt,q0,dq0)
%  Variable Description:
%     Input parameters
```

```
%       kk, cc, mm - stiffness, damping and mass matrices
%       fd - Input or forcing influence matrix
%       bcdof - Boundary condition dofs vector
%       nt - Number of time steps
%       dt - Time step size
%       q0, dq0 - Initial condition vectors
%     Output parameters
%       acc - Acceleration response
%       vel - Velocity response
%       dsp - Displacement response
%-------------------------------------------------------------------------
%  (1) initial condition
%-------------------------------------------------------------------------
[sdof,n2]=size(kk);

dsp=zeros(sdof,nt);                                  % displacement matrix
vel=zeros(sdof,nt);                                  % velocity matrix
acc=zeros(sdof,nt);                                  % acceleration matrix

dsp(:,1)=q0;                                         % initial displacement
vel(:,1)=dq0;                                        % initial velocity

alpha=0.5; beta=0.5;                                 % select the parameters
%-------------------------------------------------------------------------
%  (2) Newmark integration scheme for time integration
%-------------------------------------------------------------------------
acc(:,1)=inv(mm)*(fd(:,1)-kk*dsp(:,1)-cc*vel(:,1));
                                    % compute the initial acceleration (t=0)
ekk=kk+mm/(alpha*dt^2)+cc*beta/(alpha*dt);
                                    % compute the effective stiffness matrix
             % assign zero to dsp, vel, acc of the dofs associated with bc
for i=1:sdof
  if bcdof(i)==1
    dsp(i,1)=0;
    vel(i,1)=0;
    acc(i,1)=0;
  end
end

for it=1:nt                                     % loop for each time step
  cfm=dsp(:,it)/(alpha*dt^2)+vel(:,it)/(alpha*dt)+acc(:,it)*(0.5/alpha-1);
  cfc=dsp(:,it)*beta/(alpha*dt)+vel(:,it)*(beta/alpha-1)...
     +acc(:,it)*(0.5*beta/alpha-1)*dt;
  efd=fd(:,it)+mm*cfm+cc*cfc;         %  compute the effective force vector

  dsp(:,it+1)=inv(ekk)*efd;          % find the displacement at time t+dt
  acc(:,it+1)=(dsp(:,it+1)-dsp(:,it))/(alpha*dt^2)-
```

```
vel(:,it)/(alpha*dt)...
       -acc(:,it)*(0.5/alpha-1);     % find the acceleration at time t+dt
  vel(:,it+1)=vel(:,it)+acc(:,it)*(1-beta)*dt+acc(:,it+1)*beta*dt;
                                     % find the velocity at time t+dt
              % assign zero to acc, vel, dsp of the dofs associated with bc
  for i=1:sdof
    if bcdof(i)==1
      dsp(i,it+1)=0;
      vel(i,it+1)=0;
      acc(i,it+1)=0;
    end
  end

end

if cc(1,1)==0
  disp('The transient response results of undamping system')
else
  disp('The transient response results of damping system')
end

disp('The method is Newmark integration')
%----------------------------------------------------------------------
%    The end
%----------------------------------------------------------------------
```

以下函数用模态叠加法计算结构的动响应。

```
function  varargout=Modalresponse(varargin)
%----------------------------------------------------------------------
% Purpose:
% This function is used to calculate the modal and physical response for a
% strcutral system.
% Synopsis:
%      [eta,yim]=Modalresponse(M,K,Fu,u,t,Cy,q0,dq0,nummode)
%      [eta,yim]=Modalresponse(M,K,Fu,u,t,Cy,q0,dq0,a,b,nummode)
% Variable Description:
%      Input parameters
%      K - System siffness matrix
%      M - Sytem mass matrix
%      Fu - Force influence matrix
%      u  - Force vector
%      t  - Time of evaluation
%      Cy - Output matrix
%      q0,dq0 - Initial conditions
%      nummode - the number of extracted modes
%      a, b - Parameters for proportional damping [C]=a[M]+b[K]
%      Output parameters
```

```
%       eta - modal coordinate and velocity response
%       yim  -  physcial coordinate response
%
% Author Dr.XU Bin, Time:2006-11-29
%-----------------------------------------------------------------------
disp('')
disp('Please wait!! - The job is being performed.')
%-----------------------------------------------------------------------
 if nargin<9 |nargin>11|nargin==10
   error('Incorrect number of input arguments')
 else
    switch nargin
        case 9
 %----------------------------------------------------------------------
 % Solve the eigenvalue problem and normalized the eigenvectors
 %----------------------------------------------------------------------
  M=varargin{1};
  K=varargin{2};
  Fu=varargin{3};
  t=varargin{5};
  [n,n]=size(M);[n,m]=size(Fu);
  nstep=size(t');
  [V,D]=eig(K,M);
  [lambda,k]=sort(diag(D));      % Sort the eigenvalues and eigenvalues
  V=V(:,k);
  Factor=diag(V'*M*V);
  Vnorm=V*inv(sqrt(diag(Factor))); % Eigenvectors are normalized
  q0=varargin{7};
  dq0=varargin{8};
  nummode=varargin{9};
  eta0=Vnorm'*q0;             % Initial conditions for modal coordinates
  deta0=Vnorm'*dq0;           % both displacement and velocity
  eta=zeros(nstep,nummode);
  for i=1:nummode
         u=varargin{4};
         A=[0 1;-lambda(i) 0];
         B=[zeros(1,m);Vnorm(:,i)'*Fu];
         C=eye(2);
         D=0;
         x0=[eta0(i);deta0(i)];
         x=lsim(ss(A,B,C,D),u,t,x0);
         for j=1:nstep
             eta(j,i)=x(j,1);
         end
  end
  Cy=varargin{6};
  yim=Cy*Vnorm(:,1:nummode)*eta'; % Convert modal coordinate response to
```

```
physical coodrinate responses
varargout{1}=eta;
varargout{2}=yim;

      case 11
%----   -----------------------------------------------------------
% Solve the eigenvalue problem and normalized the eigenvectors
%-------------------------------------------------------------------
M=varargin{1};
k=varargin{2};
Fu=varargin{3};
t=varargin{5};
[n,n]=size(M);[n,m]=size(Fu);
nstep=size(t');
[V,D]=eig(K,M);
[lambda,k]=sort(diag(D));     % Sort the eigenvalues and eigenvalues
V=V(:,k);
Factor=diag(V'*M*V);
Vnorm=V*inv(sqrt(diag(Factor))); % Eigenvectors are normalized
Omega=diag(sqrt(Vnorm'*K*Vnorm));% Natural frequencies
%-----------------------------------------------------------------
% Compute modal damping matrix from the proportional damping matrix
%-----------------------------------------------------------------
a=varargin{9};
b=varargin{10};
Modamp=Vnorm'*(a*M+b*K)*Vnorm;
zeta=diag((1/2)*Modamp*inv(diag(Omega)));
if (max(zeta)>=1)
   disp( 'Warning - Maximum damping ratio is greater than or equal to 1')
   disp('You have to reselect a and b')
   pause
   disp('If you want to continue, type return key')
end
%-------------------------------------------------------------------
q0=varargin{7};
dq0=varargin{8};
nummode=varargin{11};
eta0=Vnorm'*M*q0;          % Initial conditions for modal coordinates
deta0=Vnorm'*M*dq0;        % - both displacement and velocity
eta=zeros(nstep,nummode);
for i=1:nummode

      u=varargin{4};
      A=[0 1;-lambda(i) -Modamp(i)];
      B=[zeros(1,m);Vnorm(:,i)'*Fu];
      C=eye(2);
      D=0;
```

```
        x0=[eta0(i);deta0(i)];
        x=lsim(ss(A,B,C,D),u,t,x0);
        for j=1:nstep
            eta(j,i)=x(j,1);
        end
 end
 Cy=varargin{6};
 yim=Cy*Vnorm(:,1:nummode)*eta'; % Convert modal coordinate response to
 physical coodrinate %responses
 varargout{1}=eta;
 varargout{2}=yim;
    end
end
%-------------------------------------------------------------------------
```

以下函数用于计算结构的频率响应.

```
function varargout=freqresponse(varargin)
%-------------------------------------------------------------------------
% Purpose:
% This function is used to calculate the freqency response for a strcutral
% system.
% Synopsis:
%       Hd=freqresponse(K,M,C,Omega,nummode,'displacement')
%       Hv=freqresponse(K,M,C,Omega,nummode,'velocity')
%       Ha=freqresponse(K,M,C,Omega,nummode,'acceleration')
%       [Hd,Hv,Ha]=freqresponse(K,M,C,Omega,nummode,'all')
%       Hd=freqresponse(K,M,C,Omega,nummode,'displacement',p,q)
%       Hv=freqresponse(K,M,C,Omega,nummode,'velocity',p,q)
%       Ha=freqresponse(K,M,C,Omega,nummode,'acceleration',p,q)
%       [Hd,Hv,Ha]=freqresponse(K,M,C,Omega,nummode,'all',p,q)
% Variable Description:
%       Input parameters
%       K - System siffness matrix
%       M - Sytem mass matrix
%       C - System damping matrix
%       Omega - Frequency range
%       nummode - the number of extracted modes
%       p/q   -- Serial number of the degrees of freedom
%       Output parameters
%       Hd - Displacement freqency response
%       Hv - Velocity freqency response
%       Ha - Acceleration freqency response
% Author Dr.XU Bin, Time:2006-11-29
%-------------------------------------------------------------------------
if nargin<6 |nargin>8|nargin==7
   error('Incorrect number of input arguments')
else
```

```
K=varargin{1};
M=varargin{2};
C=varargin{3};
Omega=varargin{4};
nummode=varargin{5};
nsdof=length(diag(K));
[V,D]=eig(K,M);
[lambda,k]=sort(diag(D))
V=V(:,k);
Factor=diag(V'*M*V);
Vnorm=V*inv(sqrt(diag(Factor)));
Dampr=Vnorm'*C*Vnorm;
n=length(Omega);
switch nargin

case 6
 switch varargin{6}
     case 'displacement'
    Hd1=zeros(nsdod,nsdof);
    Hd=zeros(nsdof,nsdof,n);
    for p=1:n
        for q=1:nummode
            Hd1= Hd1+Vnorm(:,q)*Vnorm(:,q)'/(lambda(q)-
            Omega(p)^2+i*Dampr(q,q)*Omega(p));
        end
        for l=1:nsdof
            for h=1:nsdof
                Hd(l,h,p)=Hd(l,h,p)+Hd1(l,h);
            end
        end
    end
    varargout{1}=Hd;
     case 'velocity'
      Hv1=zeros(nsdof,nsdof);
      Hv=zeros(nsdof,nsdof,n);
      for p=1:n
           for q=1:nummode
            Hv1= Hv1+i*Omega(p)*Vnorm(:,q)*Vnorm(:,q)'/(lambda(q)-
            Omega(p)^2+i*Dampr(q,q)*Omega(p));
           end
           for l=1:nsdof
              for h=1:nsdof
                Hv(l,h,p)=Hv(l,h,p)+Hv1(l,h);
              end
           end
      end
      varargout{1}=Hv;
```

```
    case 'acceleration'
      Ha1=zeros(nsdof,nsdof);
      Ha=zeros(nsdof,nsdof,n);
      for p=1:n
          for q=1:nummode
              Ha1= Ha1-Omega(p)^2*Vnorm(:,q)*Vnorm(:,q)'/(lambda(q)-
              Omega(p)^2+i*Dampr(q,q)*Omega(p));
          end
          for l=1:nsdof
            for h=1:nsdof
              Ha(l,h,p)=Ha(l,h,p)+Ha1(l,h);
            end
          end
      end
      varargout{1}=Ha;
    case 'all'
          Hd1=zeros(nsdof,nsdof);
          Hv1=zeros(nsdof,nsdof);
          Ha1=zeros(nsdof,nsdof);
          Hd=zeros(nsdof,nsdof,n);
          Hv=zeros(nsdof,nsdof,n);
          Ha=zeros(nsdof,nsdof,n);
       for p=1:n
          for q=1:nummode

              Hd1= Hd1+Vnorm(:,q)*Vnorm(:,q)'/(lambda(q)-
              Omega(p)^2+i*Dampr(q,q)*Omega(p));
              Hv1= Hv1+i*Omega(p)*Vnorm(:,q)*Vnorm(:,q)'/(lambda(q)-
              Omega(p)^2+i*Dampr(q,q)*Omega(p));
              Ha1= Ha1-Omega(p)^2*Vnorm(:,q)*Vnorm(:,q)'/(lambda(q)-
              Omega(p)^2+i*Dampr(q,q)*Omega(p));
          end
          for l=1:nsdof
            for h=1:nsdof
              Hd(l,h,p)=Hd(l,h,p)+Hd1(l,h);
              Hv(l,h,p)=Hv(l,h,p)+Hv1(l,h);
              Ha(l,h,p)=Ha(l,h,p)+Ha1(l,h);
            end
          end
       end
       varargout{1}=Hd;
       varargout{2}=Hv;
       varargout{3}=Ha;
end


   case 8
```

```
  nodei=varargin{7};
  nodej=varargin{8};
switch varargin{6}

 case 'displacement'
       Hd=zeros(n,1);
   for p=1:n
        for q=1:nummode
             Hd(p)= Hd(p)+Vnorm(nodei,q)*Vnorm(nodej,q)'/(lambda(q)-
             Omega(p)^2+i*Dampr(q,q)*Omega(p));
        end
   end
      varargout{1}=Hd;
 case 'velocity'
      Hv=zeros(n,1);
    for p=1:n
       for q=1:nummode
            Hv(p)= Hv(p)+i*Omega(p)*Vnorm(nodei,q)*Vnorm(nodej,q)'
            /(lambda(q)-Omega(p)^2+i*Dampr(q,q)*Omega(p));
       end
    end
      varargout{1}=Hv;

case 'acceleration'
      Ha=zeros(n,1);
      for p=1:n
          for q=1:nummode
              Ha(p)= Ha(p)-Omega(p)^2*Vnorm(nodei,q)*Vnorm(nodej,q)'
              /(lambda(q)-Omega(p)^2+i*Dampr(q,q)*Omega(p));
          end
      end
      varargout{1}=Ha;
case 'all'

       Hd=zeros(n,1);
       Hv=zeros(n,1);
       Ha=zeros(n,1);
   for p=1:n
      for q=1:nummode

          Hd(p)= Hd(p)+Vnorm(nodei,q)*Vnorm(nodej,q)'/(lambda(q)-
          Omega(p)^2+i*Dampr(q,q)*Omega(p));
          Hv(p)= Hv(p)+i*Omega(p)*Vnorm(nodei,q)*Vnorm(nodej,q)'
          /(lambda(q)-Omega(p)^2+i*Dampr(q,q)*Omega(p));
          Ha(p)= Ha(p)-Omega(p)^2*Vnorm(nodei,q)*Vnorm(nodej,q)'
          /(lambda(q)-Omega(p)^2+i*Dampr(q,q)*Omega(p));
      end
```

```
        end
        varargout{1}=Hd;
        varargout{2}=Hv;
        varargout{3}=Ha;

    end
end
end
%-----------------------------------------------------------------------------------
```

第3章 桁架结构

桁架结构由于其简单高效的优点，在工程中有广泛的应用. 在桁架结构中，各杆件连接处可以近似为铰接，杆件可以绕连接点自由转动. 这样结构中各杆件都只受轴向力，属于一维问题，此时各杆件内的位移是线性变化的，无须再对单个杆件进行单元划分，就可以得到杆件内的位移分布函数，而不是其他单元类型中构造出的近似位移分布函数. 因此对于桁架系统，有限元方法得到的是精确解.

3.1 杆单元

在一般有限元问题中，为了方便地找到适合局部区域的位移近似函数，将区域细化一般是有帮助的，因为较小区域内的位移变化规律总是比将其包含在内的较大区域内的位移变化规律简单. 但在桁架结构中，由于可以得到杆件内精确位移分布函数，所以不需要在杆件内再细分单元，可以直接以杆件为单元. 这一点和后面章节要介绍到的其他单元类型都不同，特别是要注意和梁单元的区别.

下面按照选定单元坐标系、选择位移函数和完成单元区域内刚度矩阵积分的过程来计算杆单元的刚度矩阵.

3.1.1 局部坐标系下的杆件单元刚度矩阵

杆单元本来是一维单元，如果在单元轴线与坐标轴不平行的情况下计算单元刚度矩阵，就会增加不必要的复杂性. 考虑对于一般桁架结构，无论如何选择整体坐标系，一般总会有杆件在总体坐标系中是倾斜的，所以使用如图 3.1 所示的以单元轴线为 x 轴的局部坐标来建立标准的杆件单元刚度矩阵，然后再通过坐标变化矩阵来得到实际各单元在整体坐标系下的刚度矩阵.

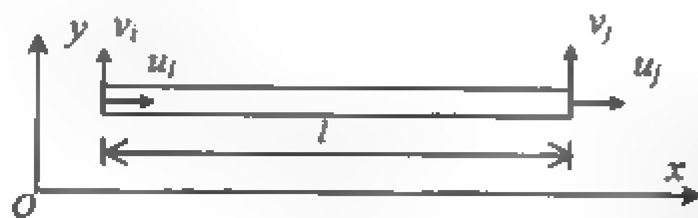


图 3.1 杆单元

实验和理论分析都证明了仅受轴向力的杆件内的位移是线性分布的. 因此与其他单元不同，不必寻找近似位移函数，而是可以直接得到杆件内的位移函数. 在上述局部坐标系下，长度为 l 的杆件内位移 u 与 x 坐标的关系可以以两端点处的位移表示如下

$$u=\left(1-\frac{x}{l}\right)u_i+\frac{x}{l}u_j \tag{3-1}$$

式中，u_i表示i端点处的位移，u_j表示j端点处的位移.

如果将i端点作为该单元的1号节点，j端点作为该单元的2号节点，相应的单元节点位移向量为$\boldsymbol{a}^e=\begin{bmatrix}u_i & u_j\end{bmatrix}^{\mathrm{T}}$. 这样可以将式(3-1)写成矩阵形式为

$$\boldsymbol{u}=\begin{bmatrix}1-\dfrac{x}{l} & 0\\ 0 & \dfrac{x}{l}\end{bmatrix}\begin{Bmatrix}u_i\\ u_j\end{Bmatrix} \tag{3-2}$$

在式(3-2)中，对于一维单元，通过位移来计算应变的关系是$\dfrac{\mathrm{d}u}{\mathrm{d}x}$，因此相应的微分算子是$\dfrac{\mathrm{d}}{\mathrm{d}x}$，由此得到

$$\boldsymbol{B}=\begin{bmatrix}-\dfrac{1}{l} & 0\\ 0 & \dfrac{1}{l}\end{bmatrix} \tag{3-3}$$

一维的杆件单元中，应力应变关系为$\sigma=E\varepsilon$，因此其弹性变量$\boldsymbol{D}$是一维的，其数值为E. 则一维杆件单元在图3.1坐标中的单元刚度矩阵为

$$\boldsymbol{K}^e=\int_V \boldsymbol{B}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{B}\mathrm{d}x=A\int_0^l \boldsymbol{B}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{B}\mathrm{d}x=\frac{EA}{l}\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix} \tag{3-4}$$

对于类似简单的单元，其刚度矩阵也通过刚度矩阵中元素的物理意义计算得到.

如果杆件轴线不是如图3.1所示同x轴平行，则杆件单元虽然是一维的，但是其位移在x和y方向上都会有投影分量，此时在其单元节点位移向量中应添加v_i和v_j，使之成为$\boldsymbol{a}^e=\begin{bmatrix}u_i & v_i & u_j & v_j\end{bmatrix}^{\mathrm{T}}$，其相应的刚度矩阵也将因此变为4阶矩阵. 所以为了保持同倾斜杆件单元的兼容，可以按照扩充后的单元节点位移向量将式(3-4)中的单元刚度矩阵扩充为4阶矩阵

$$\boldsymbol{K}^e=\frac{EA}{l}\begin{bmatrix}1 & 0 & -1 & 0\\ 0 & 0 & 0 & 0\\ -1 & 0 & 1 & 0\\ 0 & 0 & 0 & 0\end{bmatrix} \tag{3-5}$$

3.1.2 坐标转换矩阵

对于平面桁架杆单元，如图3.2所示，通过坐标系旋转即可完成从单元局部坐标系到结构整体坐标系的转换. 这一关系可以用下式表示

$$\left.\begin{aligned}x^e&=x\cos\theta+y\sin\theta\\ y^e&=-x\sin\theta+y\cos\theta\end{aligned}\right\} \tag{3-6}$$

其中上标e表示属于单元局部坐标系(以后如不作特别说明，e作上标或下标时，均代表单元(element)，用正体).

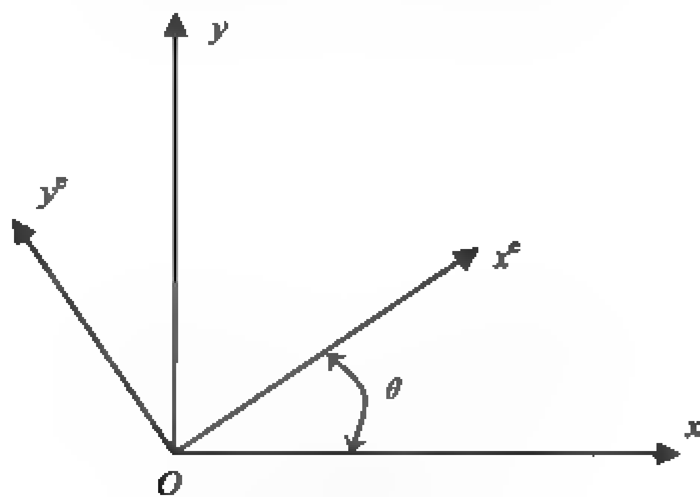


图 3.2　平面桁架杆单元

式(3-6)可以写成矩阵形式为

$$\begin{Bmatrix} x^e \\ y^e \end{Bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{Bmatrix} x \\ y \end{Bmatrix} = \boldsymbol{\lambda} \begin{Bmatrix} x \\ y \end{Bmatrix} \tag{3-7}$$

在得到了坐标旋转矩阵 $\boldsymbol{\lambda}$ 之后，可知单元局部坐标系下的单元位移向量 $\boldsymbol{a}^e$ 与系统整体坐标系下的单元位移向量 $\boldsymbol{a}$ 的关系为

$$\boldsymbol{a}^e = \begin{Bmatrix} u_i^e \\ v_i^e \\ u_j^e \\ v_j^e \end{Bmatrix} = \begin{bmatrix} \boldsymbol{\lambda} & 0 \\ 0 & \boldsymbol{\lambda} \end{bmatrix} \begin{Bmatrix} u_i \\ v_i \\ u_j \\ v_j \end{Bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & \cos\theta & \sin\theta \\ 0 & 0 & -\sin\theta & \cos\theta \end{bmatrix} \begin{Bmatrix} u_i \\ v_i \\ u_j \\ v_j \end{Bmatrix} = \boldsymbol{T}\boldsymbol{a} \tag{3-8}$$

式中，$\boldsymbol{T}$ 称为转换矩阵.

下面分析如何利用转换矩阵将单元刚度矩阵从单元局部坐标系转换到结构整体坐标系下的表达式.

在单元局部坐标系中建立一个单元的平衡方程，得到

$$\boldsymbol{K}^e\boldsymbol{a}^e = \boldsymbol{P}^e \tag{3-9}$$

为了得到在结构整体坐标系的相应方程，需要将式(3-9)中的 $\boldsymbol{a}^e$ 和 $\boldsymbol{P}^e$ 分别替换为 $\boldsymbol{a}$ 和 $\boldsymbol{P}$. 对于单元位移向量，已经给出 $\boldsymbol{a}^e = \boldsymbol{T}\boldsymbol{a}$ ；对于单元载荷向量，相应的也有 $\boldsymbol{P}^e = \boldsymbol{T}\boldsymbol{P}$ ，分别代入式(3-9)中得

$$\boldsymbol{K}^e\boldsymbol{T}\boldsymbol{a} = \boldsymbol{T}\boldsymbol{P} \tag{3-10}$$

在两边同时左乘 $\boldsymbol{T}^{-1}$ ，可得

$$\boldsymbol{T}^{-1}\boldsymbol{K}^e\boldsymbol{T}\boldsymbol{a} = \boldsymbol{P} \tag{3-11}$$

记 $\boldsymbol{T}^{-1}\boldsymbol{K}^e\boldsymbol{T}$ 为 $\boldsymbol{K}$ ，就得到了在结构整体坐标系下与式(3-9)相同形式的平衡方程

$$\boldsymbol{K}\boldsymbol{a} = \boldsymbol{P} \tag{3-12}$$

从上述过程中可以得知，两种坐标系下的单元刚度矩阵的关系为

$$\boldsymbol{K} = \boldsymbol{T}^{-1}\boldsymbol{K}^e\boldsymbol{T} \tag{3-13}$$

可以验证，坐标转换矩阵总是正交矩阵，即有 $\boldsymbol{T}' = \boldsymbol{T}^{-1}$ ，所有在实际应用中，一般使用式(3-13)的等价形式

$$\boldsymbol{K} = \boldsymbol{T}'\boldsymbol{K}^e\boldsymbol{T} \tag{3-14}$$

将式(3-5)和 $\boldsymbol{T}$ 带入式(3-14)，可以得到

$$\boldsymbol{K}=\frac{EA}{l}\begin{bmatrix}\cos^2\theta & \cos\theta\sin\theta & -\cos^2\theta & -\cos\theta\sin\theta\\ \cos\theta\sin\theta & \sin^2\theta & -\cos\theta\sin\theta & -\sin^2\theta\\ -\cos^2\theta & -\cos\theta\sin\theta & \cos^2\theta & \cos\theta\sin\theta\\ -\cos\theta\sin\theta & -\sin^2\theta & \cos\theta\sin\theta & \sin^2\theta\end{bmatrix} \tag{3-15}$$

可以验证，先在单元局部坐标系下推导单元刚度矩阵，再使用转换矩阵得到的式(3-15)的结果，和直接在结构整体坐标系下得到的一般杆件单元刚度矩阵是一致的.

3.1.3 单元质量矩阵

在本书第 1 章中已经介绍过，存在两种单元质量矩阵，其中协调单元质量矩阵可以按照下式计算

$$\boldsymbol{M}^e=\int_{V_e}\rho\boldsymbol{N}^T\boldsymbol{N}\mathrm{d}V \tag{3-16}$$

其计算过程与式(3-11)中单元刚度矩阵的计算过程类似，在这里不再叙述而是直接给出结果

$$\boldsymbol{M}^e=\frac{\rho Al}{4}\begin{bmatrix}2 & 0 & 1 & 0\\ 0 & 2 & 0 & 1\\ 1 & 0 & 2 & 0\\ 0 & 1 & 0 & 2\end{bmatrix} \tag{3-17}$$

式中，A 为杆件的截面积.

平面杆单元的集中质量矩阵为

$$\boldsymbol{M}^e=\frac{\rho Al}{2}\boldsymbol{I}_{4\times4} \tag{3-18}$$

式中，$\boldsymbol{I}_{4\times4}$ 表示 4 阶单位矩阵.

3.1.4 三维杆单元

上面两节内容是针对平面桁架结构的. 对于空间桁架结构，应使用三维杆单元. 在推导三维杆单元的过程中，继续使用单元局部坐标系依旧是有利的. 由于杆单元本身的一维特点，在使用杆件轴线作为单元局部坐标系 x 轴的情况下，三维杆单元与二维杆单元是基本相同的. 三维杆单元中增加的单元节点位移 w_1 和 w_2 在此情况下实际上和另一组位移 v_1 和 v_2 一样总为零，杆件内的位移分布通过对 u_1 和 u_2 进行插值即可得到. 这样两种单元的推导过程和结果都是一样的. 只是为了和在结构整体坐标系下的一般杆件单元的刚度矩阵相统一，对三维杆单元还是需要在单元位移向量中引入 w_1、w_2、v_1 和 v_2，同时也相应地将单元刚度矩阵扩充为 6 阶.

$$K^e = \frac{EA}{l}\begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3-19}$$

三维杆单元的集中质量矩阵为

$$M^e = \frac{\rho A l}{2} I_{6\times 6} \tag{3-20}$$

协调质量矩阵为

$$M^e = \frac{\rho A l}{6}\begin{bmatrix} 2 & 0 & 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 & 1 & 0 \\ 0 & 0 & 2 & 0 & 0 & 1 \\ 1 & 0 & 0 & 2 & 0 & 0 \\ 0 & 1 & 0 & 0 & 2 & 0 \\ 0 & 0 & 1 & 0 & 0 & 2 \end{bmatrix} \tag{3-21}$$

三维杆单元的单元局部坐标系和结构整体坐标系之间依旧是坐标系旋转的关系，其转换矩阵为

$$\lambda = \begin{bmatrix} \cos(x,x^e) & \cos(x,y^e) & \cos(x,z^e) \\ \cos(y,x^e) & \cos(y,y^e) & \cos(y,z^e) \\ \cos(z,x^e) & \cos(z,y^e) & \cos(z,z^e) \end{bmatrix} \tag{3-22}$$

$$T = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \tag{3-23}$$

其中(x,x^e)表示结构整体坐标系 x 轴与单元局部坐标系 x 轴间的角度；(x,y^e)表示结构整体坐标系 x 轴与单元局部坐标系 y 轴间的角度；其余以此类推.

3.2 算　例

下面通过计算如图 3.3 所示桁架的固有频率来说明桁架结构有限元程序的一般过程.

3.2.1 问题介绍

该桁架由 5 个节点和 7 个杆件构成. 各杆件的截面积统一为 A=1e-4m^3，弹性模量为 E=2.1e11Pa，质量密度为 $\rho = 7\,300\text{kg/m}^3$. 结构在 1 号节点处有水平和垂直方向上的位移约束，在 3 号节点上有垂直方向上的位移约束. 结构各节点位置如图 3.3 所示.

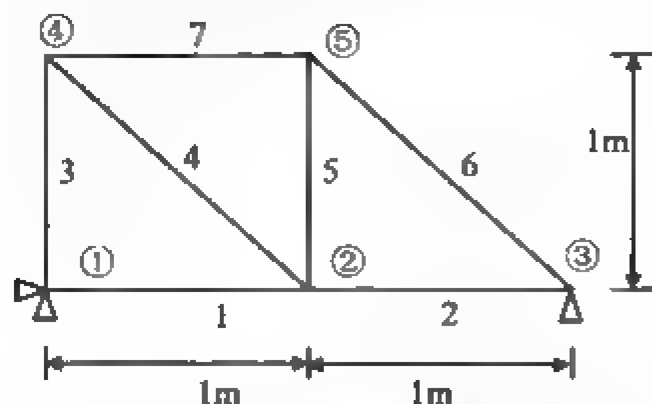


图3.3 平面桁架结构

该结构为平面结构，所以计算其固有频率的时候选择二维杆件单元即可. 如果选择三维杆件单元，计算出的结果中会增加5阶低阶频率. 读者可以思考为什么是增加了5阶？为什么这些频率是不需要的？

3.2.2 MATLAB程序及说明

计算该问题的MATLAB程序如下：

```
E=2.1e11;
A=1e-4;
density=7.3e3;
node_number=5;
element_number=7;

nc=[0,0;1,0;2,0;0,1;1,1];
%nc:node_coordinate

en=[1,2;2,3;1,4;2,4;2,5;3,5;4,5];
%en:element_node
ed(1:node_number,1:2)=1;
%ed:element_displacement
constraint=[1,1;1,2;3,2];

for loopi=1:length(constraint);
    ed(constraint(loopi,1),constraint(loopi,2))=0;
end
dof=0;
for loopi=1:node_number
     for loopj=1:2
        if ed(loopi,loopj)~=0
            dof=dof+1;
            ed(loopi,loopj)=dof;
        end
    end
end
```

```
%el:length of link element
ek=E*A*[1 0 -1 0;0 0 0 0;-1 0 1 0;0 0 0 0];
%ek:element_stiffness_matrix
em=(density*A)/2*eye(4);
%em:element mass_matrix
k(1:dof,1:dof)=0;
%k:structural_stiffness_matrix
m=k;
%m:structural_mass matrix,same size with k
theta(1:7)=0;
el(1:7)=0;
e2s(1:4)=0;
% e2s: index of transform the element displament number to structural
for loopi=1:element_number
    for zi=1:2
        e2s((zi-1)*2+1)=ed(en(loopi,zi),1);
        e2s((zi-1)*2+2)=ed(en(loopi,zi),2);
    end

    el(loopi)=sqrt((nc(en(loopi,1),1)-nc(en(loopi,2),1))^2
    +(nc(en(loopi,1),2)-nc(en(loopi,2),2))^2);
    theta(loopi)=asin((nc(en(loopi,1),2)-nc(en(loopi,2),2))/el(loopi));
    lmd=[cos(theta(loopi)) sin(theta(loopi)); -sin(theta(loopi))
    cos(theta(loopi))];
    t=[lmd zeros(2); zeros(2) lmd];
    dk=t'*ek*t/el(loopi);
    dm=t'*em*t*el(loopi);

    for jx=1:4
        for jy=1:4
            if(e2s(jx)*e2s(jy)~=0)
                k(e2s(jx),e2s(jy))=k(e2s(jx),e2s(jy))+dk(jx,jy);
                m(e2s(jx),e2s(jy))=m(e2s(jx),e2s(jy))+dm(jx,jy);
            end
        end
    end
end

[v,d] = eig(k,m);
%d:eigenvalue
%v:eigenvector

frequency=sqrt(diag(d))/(2*pi);

[frequency,indexf]=sort(frequency);
d=d(:,indexf);
```

在这个程序中，采取的是通过式(3-14)由标准的单元局部坐标系下的单元刚度矩阵计算各单元在整体坐标系下的刚度矩阵的方法. 实际上也可以对各单元直接使用式(3-15)给出的结果. 因为在本例中，只有 7 个单元，所以两种方法差别不大. 对于大型结构，就有必要使用后一种方法.

程序中用到的 eye 函数用来生产单元矩阵. zeros 函数用来生成元素全为零的矩阵. 在本例中，由于系统自由度数目仅为 7，总体刚度矩阵和质量矩阵都很小，所以没有使用稀疏矩阵来进行保存. 对于大型桁架结构，可以考虑在程序中使用稀疏矩阵.

程序的最后两条语句用来对频率进行排序. 因为 eig 函数求得的特征值并不是已经排序好的. 所以有必要对 frequency 变量进行排序，并且根据其顺序的改变，也相应地改变特征向量数组的排序.

3.2.3 计算结果

程序计算结果和使用商业化有限元软件 Abaqus 所得到的结果列在表 3.1 中. 可以看出，两者结果是完全一致的.

表 3.1 算例固有频率

	Frequency (Hz) (MATLAB)	Frequency by (Hz) (Abaqus)
1	237.5939	237.59
2	285.2482	285.25
3	570.4911	570.49
4	746.4871	746.49
5	966.6246	966.62
6	1 007.8682	1007.9
7	1 175.3168	1175.3

可以使用 line 命令来绘图显示此桁架结构的模态.

```
mode=1;
for loopi=1:element_number
    x=[nc(en(loopi,1),1) nc(en(loopi,2),1)];
    y=[nc(en(loopi,1),2) nc(en(loopi,2),2)];
    line(x,y)
    mx(1:2)=0;
    my(1:2)=0;
    for loopj=1:2
        if ed(en(loopi,loopj),1)~=0
            mx(loopj)=x(loopj)+v(ed(en(loopi,loopj),1),mode)/8;
        else
            mx(loopj)=x(loopj);
        end
```

```
        if ed(en(loopi,loopj),2)~=0
           my(loopj)=y(loopj)+v(ed(en(loopi,loopj),2),mode)/8;
        else
           my(loopj)=y(loopj);
        end
    end
    line(mx,my,'LineStyle',':')
end
```

通过 line 命令绘制直线并不复杂，上述程序的主要部分是在于根据特征向量得到结构变形后各节点的位置坐标，所得第一阶模态如图 3.4 所示，其中实线部分表示原结构以作对比.

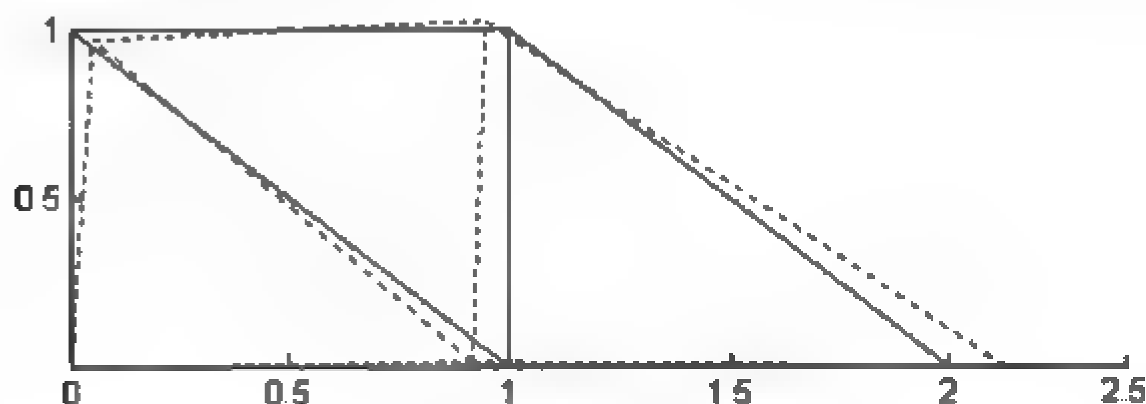


图 3.4 桁架结构第一阶模态

通过调整上述程序中 mode 的数值，可以很方便地得到其他几阶模态的图形. 图 3.5～图 3.7 分别给出了第二阶到第四阶模态.

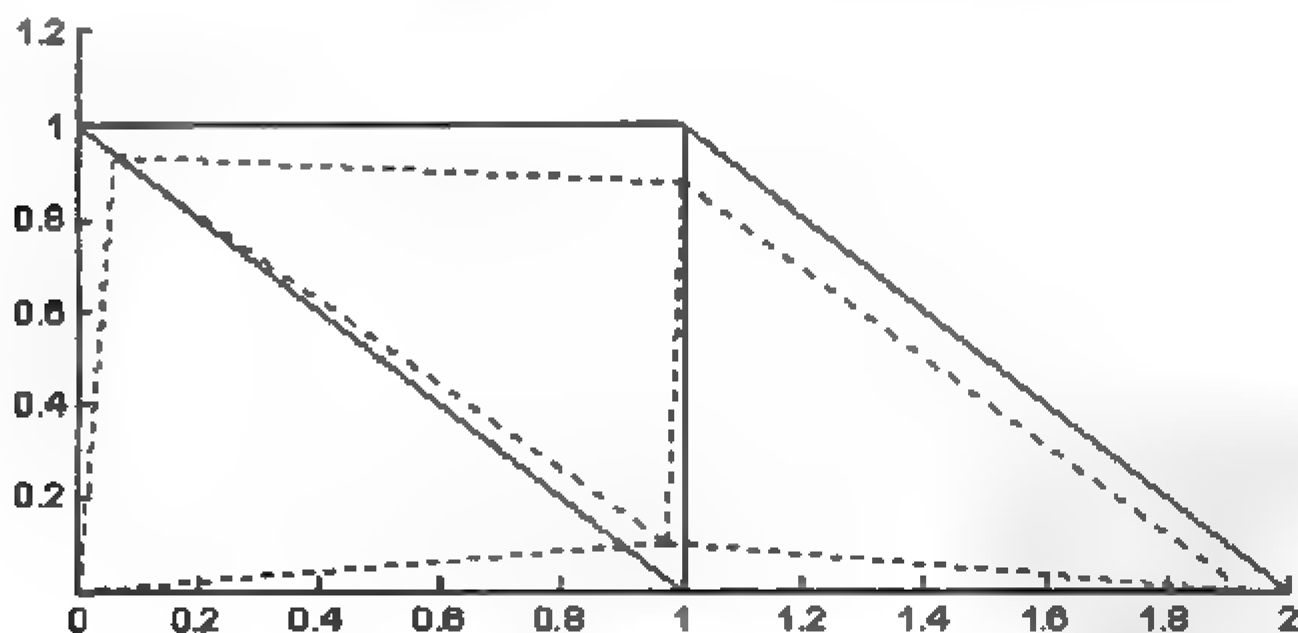


图 3.5 桁架结构第二阶模态

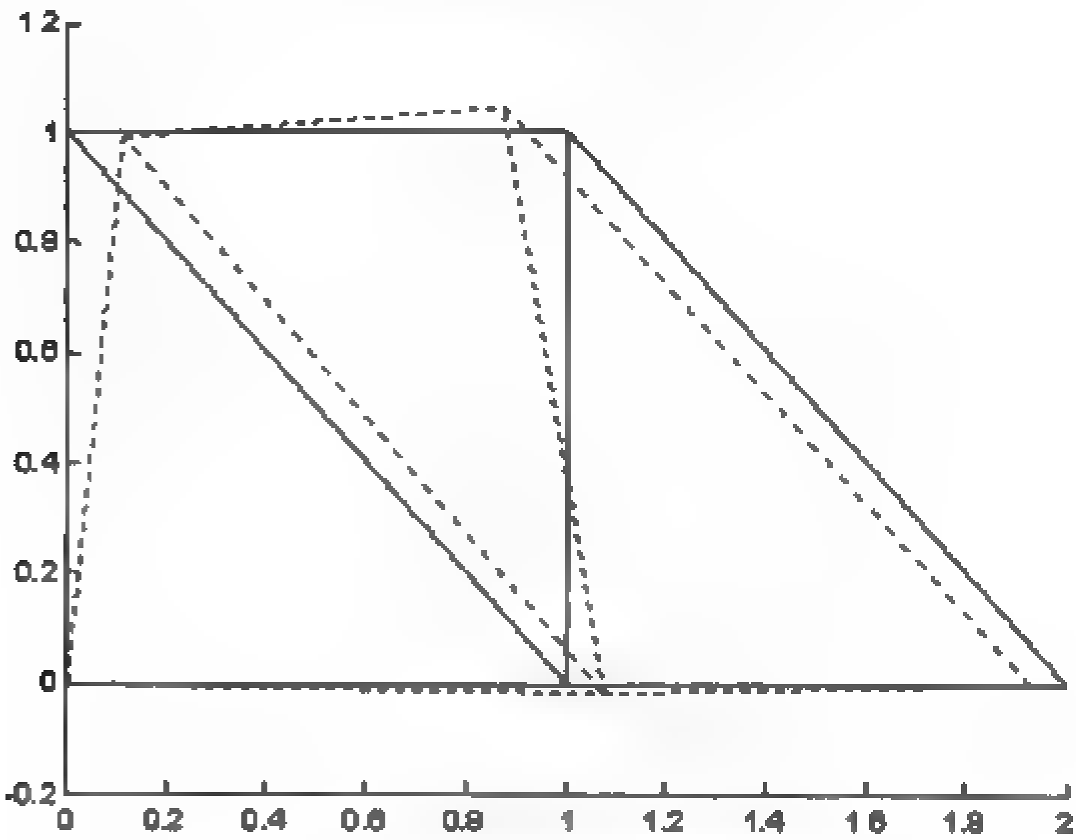


图 3.6 桁架结构第三阶模态

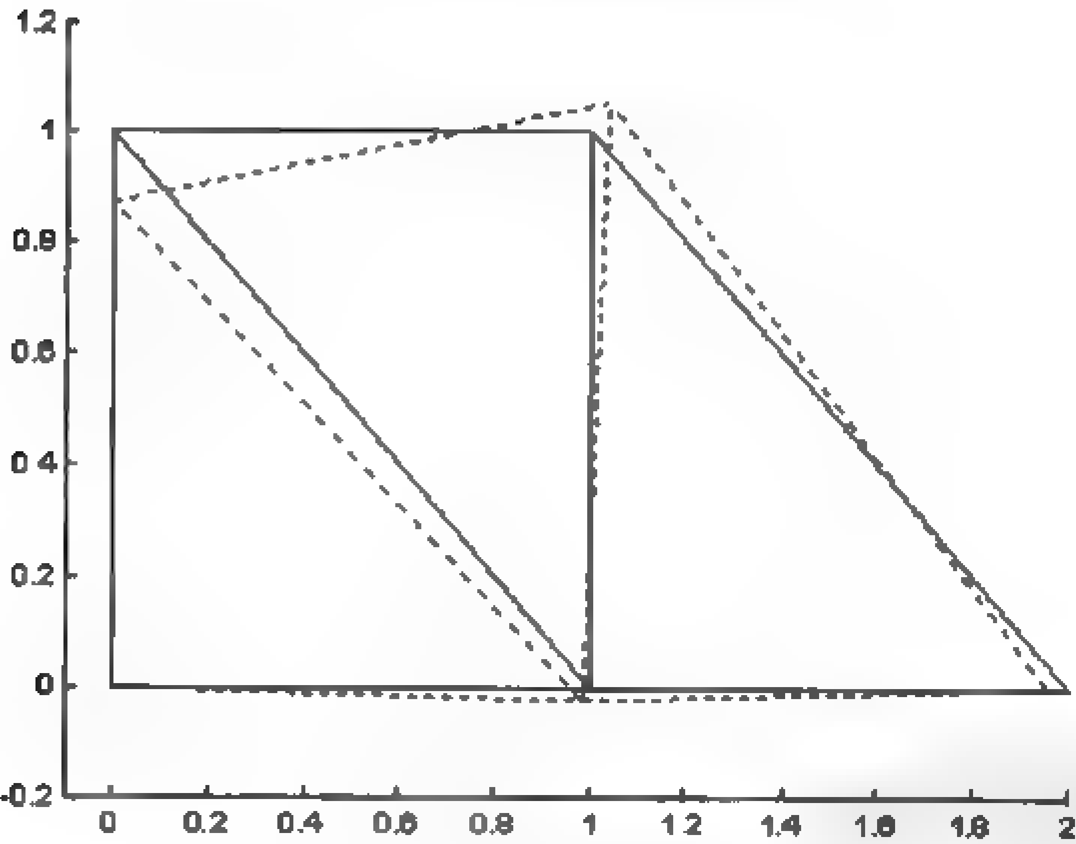


图 3.7 桁架结构第四阶模态

第4章 等参单元

在有限元法的计算中，随着单元节点数目的增加，插值函数的阶次也相应增高，使得求解精度提高，在对一个给定问题的求解域，可用较少的单元数获得满足精度要求的解答．但是，实际进行有限元建模时，用较少的形状规则的单元来离散几何形状复杂的求解域时常会产生困难．若用边界为曲线或曲面的单元，则可使此问题得到解决，因此需要建立将形状规则的单元变换为边界为曲线或曲面的单元的方法．在有限元法中对应此问题所采用的变换方法是等参变换，即单元几何形状的变换和单元内的场函数采用相同数目的节点及相同的插值函数进行变换．

等参变换的实质是从一个坐标系到另一个坐标系的数学映射．前者称为自然坐标系(或局部坐标系)，后者称为物理坐标系(或总体坐标系)．将自然坐标系中几何形状规则的单元转换物理(笛卡儿)坐标系中几何形状扭曲的单元，需要建立一个坐标变换．

$$\begin{Bmatrix} x \\ y \\ z \end{Bmatrix} = f\left(\begin{Bmatrix} \xi \\ \eta \\ \varsigma \end{Bmatrix}\right) \quad 或 \quad \begin{Bmatrix} x \\ y \\ z \end{Bmatrix} = f\left(\begin{Bmatrix} L_1 \\ L_2 \\ L_3 \\ L_4 \end{Bmatrix}\right)$$

图4.1所示为等参变换的例子．

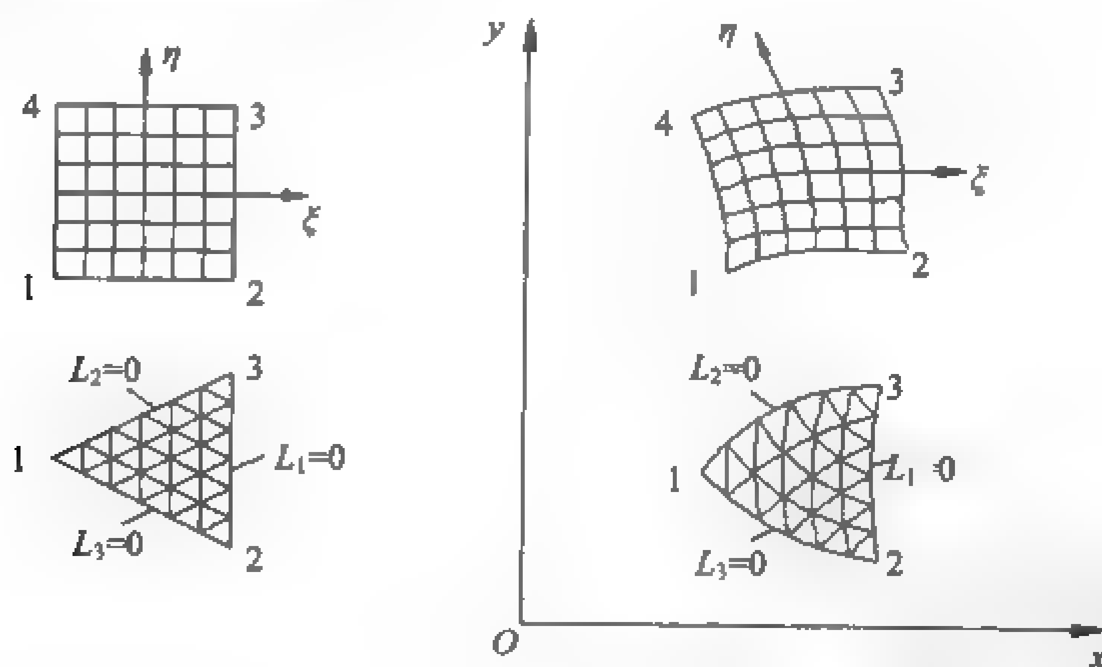


图4.1 二维单元的等参变换

4.1 一维单元

一维线性单元的等参变换如图4.2所示．

单元有两个节点，在如图4.2(a)所示的自然坐标系中为 $\xi_1=-1$ 和 $\xi_2=1$，在如图4.2(b)所示的物理坐标系中为 x_1 和 x_2，相应的节点位移为 u_1 和 u_2．在自然坐标系下，单元的形

函数可写为

$$
\begin{aligned}
N_1(\xi) &= \frac{1}{2}(1-\xi) \\
N_2(\xi) &= \frac{1}{2}(1+\xi)
\end{aligned} \tag{4-1}
$$



(a) 自然坐标系中的情况　　　(b) 物理坐标系中的情况

图 4.2　一维线性单元的等参变换

利用此形函数，在自然坐标系下单元内的任一点 $-1<\xi<1$ 可以映射到物理坐标系下，有

$$x = N_1(\xi)x_1 + N_2(\xi)x_2 \tag{4-2}$$

用同一形函数对单元的位移变量 u 插值，有

$$u = N_1(\xi)u_1 + N_2(\xi)u_2 \tag{4-3}$$

这种采用相同的形函数进行坐标变换和节点变量插值，并且用相同节点数的单元称为等参单元，这种变换称为等参变换．如果变换用的节点数大于函数插值用的节点数，则称为超参变换；反之，如果变换用的节点数小于函数插值用的节点数，则称为次参变换．

为了便于在规格化的自然坐标系下进行单元特性矩阵的计算，需要建立两个坐标系间导数、面积微元、体积微元的变换关系．

对于一维单元

$$\frac{\mathrm{d}u}{\mathrm{d}x} = \frac{\mathrm{d}N_1(\xi)}{\mathrm{d}x}u_1 + \frac{\mathrm{d}N_2(\xi)}{\mathrm{d}x}u_2 = \frac{\mathrm{d}N_1(\xi)}{\mathrm{d}\xi}\frac{\mathrm{d}\xi}{\mathrm{d}x}u_1 + \frac{\mathrm{d}N_2(\xi)}{\mathrm{d}\xi}\frac{\mathrm{d}\xi}{\mathrm{d}x}u_2 \tag{4-4}$$

由式(4-2)，有

$$\frac{\mathrm{d}x}{\mathrm{d}\xi} = \frac{\mathrm{d}N_1(\xi)}{\mathrm{d}\xi}x_1 + \frac{\mathrm{d}N_2(\xi)}{\mathrm{d}\xi}x_2 = \frac{1}{2}(x_2 - x_1) = \frac{1}{2}l_i \tag{4-5}$$

式中，$l_i = x_2 - x_1$，是单元的长度．

式(4-5)称为 Jacobi 值，可表示为 $J = \mathrm{d}x/\mathrm{d}\xi$．将其代入式(4-4)，得

$$\frac{\mathrm{d}u}{\mathrm{d}x} = -\frac{1}{x_2 - x_1}u_1 + \frac{1}{x_2 - x_1}u_2 \tag{4-6}$$

因而对应于物理坐标系，形函数的导数为

$$
\begin{aligned}
\frac{\mathrm{d}N_1(\xi)}{\mathrm{d}x} &= -\frac{1}{x_2 - x_1} = -\frac{1}{l_i} \\
\frac{\mathrm{d}N_2(\xi)}{\mathrm{d}x} &= \frac{1}{x_2 - x_1} = \frac{1}{l_i}
\end{aligned} \tag{4-7}
$$

对于如图 4.3 所示的一维二次等参单元，形函数为

$$N_1(\xi) = \frac{\xi(\xi - 1)}{2}，\quad N_2(\xi) = 1 - \xi^2，\quad N_3(\xi) = \frac{\xi(\xi + 1)}{2} \tag{4-8}$$

用该形函数对单元的位移变量u插值，有

$$u = N_1(\xi)u_1 + N_2(\xi)u_2 + N_3(\xi)u_3 \tag{4-9}$$

单元从自然坐标系到物理坐标系映射为

$$x = N_1(\xi)x_1 + N_2(\xi)x_2 + N_3(\xi)x_3 \tag{4-10}$$

Jacobi 值为

$$J = \frac{\mathrm{d}x}{\mathrm{d}\xi} = \sum_{i=1}^{3}\frac{\mathrm{d}H_i(\xi)}{\mathrm{d}\xi}x_i = (\xi - 0.5)x_1 - 2\xi x_2 + (\xi + 0.5)x_3 \tag{4-11}$$

若物理坐标系中的节点x_2位于节点x_1与x_3的中间，即有$x_2 = \frac{x_1 + x_3}{2}$，Jacobi 值变为

$$J = \frac{1}{2}(x_3 - x_1) = \frac{1}{2}l_i \tag{4-12}$$

对应于物理坐标系，形函数的导数为

$$\begin{aligned}\frac{\mathrm{d}N_1(\xi)}{\mathrm{d}x} &= \frac{1}{J}\frac{\mathrm{d}N_1(\xi)}{\mathrm{d}\xi} = \frac{1}{l_i}(2\xi - 1)\\ \frac{\mathrm{d}N_2(\xi)}{\mathrm{d}x} &= \frac{1}{J}\frac{\mathrm{d}N_2(\xi)}{\mathrm{d}\xi} = -\frac{4\xi}{l_i}\\ \frac{\mathrm{d}N_3(\xi)}{\mathrm{d}x} &= \frac{1}{J}\frac{\mathrm{d}N_3(\xi)}{\mathrm{d}\xi} = \frac{1}{l_i}(2\xi + 1)\end{aligned} \tag{4-13}$$

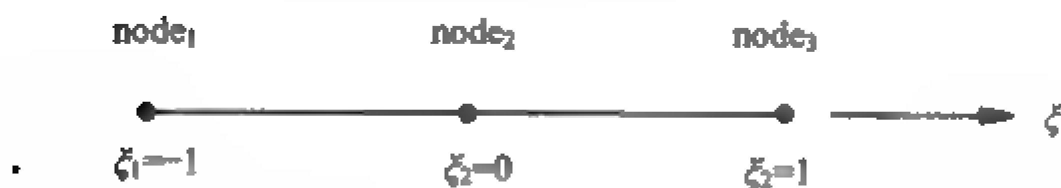
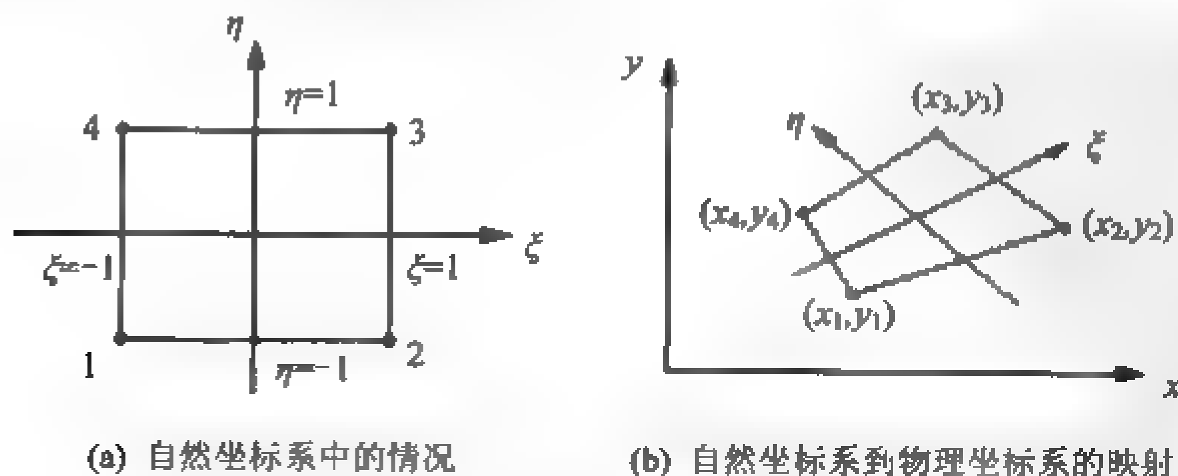


图 4.3 一维二次单元

4.2 四边形单元

四边形单元可由自然坐标系中的矩形单元映射而成，如图 4.4 所示．如图 4.4(a)所示，在自然坐标系下，单元是规则化的(即$-1 \leqslant \xi \leqslant 1$，$-1 \leqslant \eta \leqslant 1$)．



(a) 自然坐标系中的情况　(b) 自然坐标系到物理坐标系的映射

图 4.4 四边形单元

当自然坐标系中单元取为双线性单元时，单元的形函数定义如下

$$
\begin{aligned}
N_1(\xi,\eta) &= \frac{1}{4}(1-\xi)(1-\eta) \\
N_2(\xi,\eta) &= \frac{1}{4}(1+\xi)(1-\eta) \\
N_3(\xi,\eta) &= \frac{1}{4}(1+\xi)(1+\eta) \\
N_4(\xi,\eta) &- \frac{1}{4}(1-\xi)(1+\eta)
\end{aligned} \tag{4-14}
$$

单元从自然坐标系到物理坐标系的映射为

$$x = \sum_{i=1}^{4} N_i(\xi,\eta) x_i \tag{4-15}$$

$$y = \sum_{i=1}^{4} N_i(\xi,\eta) y_i \tag{4-16}$$

在进行映射变换时，要求单元两个坐标系下的节点编号要对应.

单元的节点变量用形函数进行插值，有

$$u = \sum_{i=1}^{4} N_i(\xi,\eta) u_i \tag{4-17}$$

对应于自然坐标系，形函数的导数为

$$
\begin{aligned}
\frac{\partial N_i(\xi,\eta)}{\partial \xi} &= \frac{\partial N_i(\xi,\eta)}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial N_i(\xi,\eta)}{\partial y}\frac{\partial y}{\partial \xi} \\
\frac{\partial N_i(\xi,\eta)}{\partial \eta} &= \frac{\partial N_i(\xi,\eta)}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial N_i(\xi,\eta)}{\partial y}\frac{\partial y}{\partial \eta}
\end{aligned} \tag{4-18}
$$

写成矩阵形式为

$$
\begin{Bmatrix} \dfrac{\partial}{\partial \xi} \\ \dfrac{\partial}{\partial \eta} \end{Bmatrix} N_i(\xi,\eta) = \begin{bmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial y}{\partial \xi} \\ \dfrac{\partial x}{\partial \eta} & \dfrac{\partial y}{\partial \eta} \end{bmatrix} \begin{Bmatrix} \dfrac{\partial}{\partial x} \\ \dfrac{\partial}{\partial y} \end{Bmatrix} N_i(\xi,\eta) = \boldsymbol{J} \begin{Bmatrix} \dfrac{\partial}{\partial x} \\ \dfrac{\partial}{\partial y} \end{Bmatrix} N_i(\xi,\eta) \tag{4-19}
$$

式中，$\boldsymbol{J}$ 称为 Jacobi 矩阵，且有

$$
\boldsymbol{J} = \begin{bmatrix} J_{11} & J_{21} \\ J_{12} & J_{22} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial y}{\partial \xi} \\ \dfrac{\partial x}{\partial \eta} & \dfrac{\partial y}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \sum\limits_{i=1}^{4} \dfrac{\partial N_i(\xi,\eta)}{\partial \xi} x_i & \sum\limits_{i=1}^{4} \dfrac{\partial N_i(\xi,\eta)}{\partial \xi} y_i \\ \sum\limits_{i=1}^{4} \dfrac{\partial N_i(\xi,\eta)}{\partial \eta} x_i & \sum\limits_{i=1}^{4} \dfrac{\partial N_i(\xi,\eta)}{\partial \eta} y_i \end{bmatrix} \tag{4-20}
$$

对应于物理坐标系，形函数的导数为

$$
\begin{aligned}
\frac{\partial N_i(\xi,\eta)}{\partial x} &= \frac{\partial N_i(\xi,\eta)}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial N_i(\xi,\eta)}{\partial \eta}\frac{\partial \eta}{\partial x} \\
\frac{\partial N_i(\xi,\eta)}{\partial y} &= \frac{\partial N_i(\xi,\eta)}{\partial \xi}\frac{\partial \xi}{\partial y} + \frac{\partial N_i(\xi,\eta)}{\partial \eta}\frac{\partial \eta}{\partial y}
\end{aligned} \tag{4-21}
$$

写成矩阵形式为

$$\begin{Bmatrix}\dfrac{\partial}{\partial x}\\ \dfrac{\partial}{\partial y}\end{Bmatrix}N_i(\xi,\eta)=\begin{bmatrix}\dfrac{\partial\xi}{\partial x} & \dfrac{\partial\eta}{\partial x}\\ \dfrac{\partial\xi}{\partial y} & \dfrac{\partial\eta}{\partial y}\end{bmatrix}\begin{Bmatrix}\dfrac{\partial}{\partial\xi}\\ \dfrac{\partial}{\partial\eta}\end{Bmatrix}N_i(\xi,\eta)=\boldsymbol{R}\begin{Bmatrix}\dfrac{\partial}{\partial\xi}\\ \dfrac{\partial}{\partial\eta}\end{Bmatrix}N_i(\xi,\eta) \tag{4-22}$$

比较式(4-19)和式(4-22)，可知

$$\boldsymbol{R}=\boldsymbol{J}^{-1}=\begin{bmatrix}R_{11} & R_{21}\\ R_{12} & R_{22}\end{bmatrix} \tag{4-23}$$

由 $\mathrm{d}\xi$ 和 $\mathrm{d}\eta$ 在物理坐标系中形成的面积微元为

$$\mathrm{d}A=\mathrm{d}\boldsymbol{\xi}\times\mathrm{d}\boldsymbol{\eta} \tag{4-24}$$

而

$$\begin{aligned}\mathrm{d}\boldsymbol{\xi}&=\frac{\partial x}{\partial\xi}\mathrm{d}\xi\boldsymbol{i}+\frac{\partial y}{\partial\xi}\mathrm{d}\xi\boldsymbol{j}\\ \mathrm{d}\boldsymbol{\eta}&=\frac{\partial x}{\partial\eta}\mathrm{d}\eta\boldsymbol{i}+\frac{\partial y}{\partial\eta}\mathrm{d}\eta\boldsymbol{j}\end{aligned} \tag{4-25}$$

式中，$\boldsymbol{i}$、$\boldsymbol{j}$ 是物理坐标系的单元坐标向量。将式(4-25)代入式(4-24)，得

$$\mathrm{d}A=\begin{vmatrix}\dfrac{\partial x}{\partial\xi} & \dfrac{\partial y}{\partial\xi}\\ \dfrac{\partial x}{\partial\eta} & \dfrac{\partial y}{\partial\eta}\end{vmatrix}\mathrm{d}\xi\,\mathrm{d}\eta=|\boldsymbol{J}|\mathrm{d}\xi\,\mathrm{d}\eta \tag{4-26}$$

在实际应用中，常用到八节点的四边形等参单元和九节点的四边形等参单元，如图 4.5 和图 4.6 所示，单元的形函数见表 4.1。

表 4.1　四边形等参单元的形函数

八节点的四边形等参单元	九节点的四边形等参单元
$N_1=\frac{1}{4}(1-\xi)(1-\eta)(-1-\xi-\eta)$	$N_1=\frac{1}{2}(\xi^2-\xi)(\eta^2-\eta)$
$N_2=\frac{1}{4}(1+\xi)(1-\eta)(-1+\xi-\eta)$	$N_2=\frac{1}{2}(\xi^2+\xi)(\eta^2-\eta)$
$N_3=\frac{1}{4}(1+\xi)(1+\eta)(-1+\xi+\eta)$	$N_3=\frac{1}{2}(\xi^2+\xi)(\eta^2+\eta)$
$N_4=\frac{1}{4}(1-\xi)(1+\eta)(-1-\xi+\eta)$	$N_4=\frac{1}{2}(\xi^2-\xi)(\eta^2+\eta)$
$N_5=\frac{1}{2}(1-\xi^2)(1-\eta)$	$N_5=\frac{1}{2}(1-\xi^2)(\eta^2-\eta)$
$N_6=\frac{1}{2}(1+\xi)(1-\eta^2)$	$N_6=\frac{1}{2}(\xi^2+\xi)(1-\eta^2)$
$N_7=\frac{1}{2}(1-\xi^2)(1+\eta)$	$N_7=\frac{1}{2}(1-\xi^2)(\eta^2+\eta)$
$N_8=\frac{1}{2}(1-\xi)(1-\eta^2)$	$N_8=\frac{1}{2}(\xi^2-\xi)(1-\eta^2)$
	$N_9=\frac{1}{2}(1-\xi^2)(1-\eta^2)$

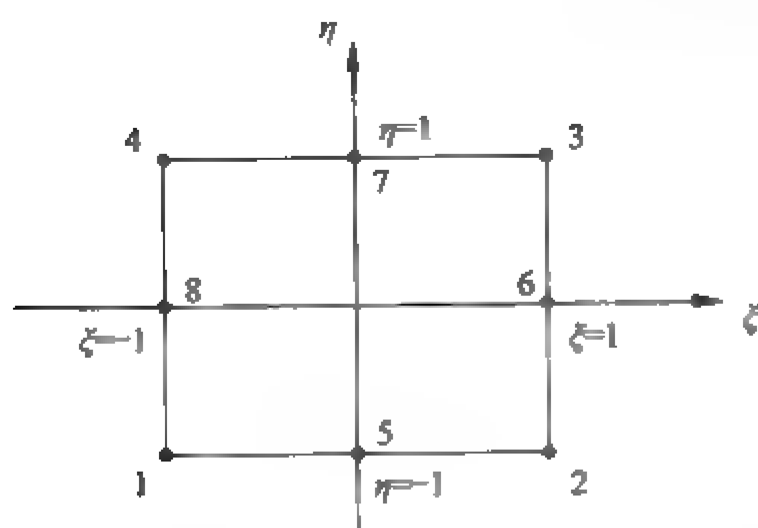


图 4.5 八节点的四边形等参单元

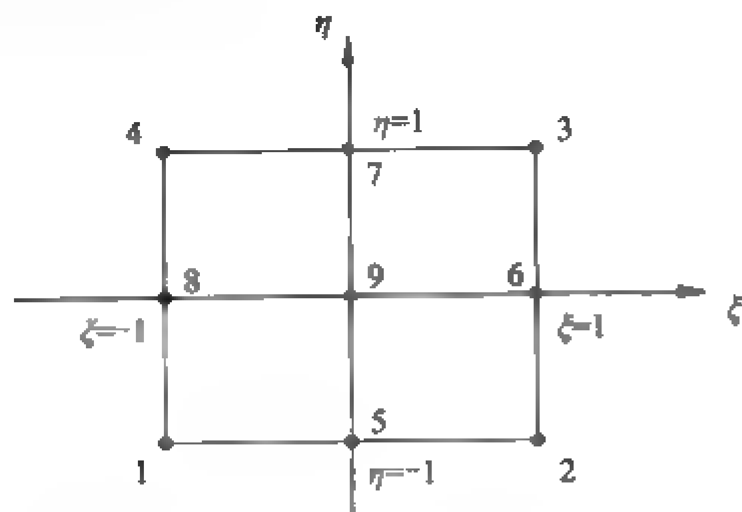
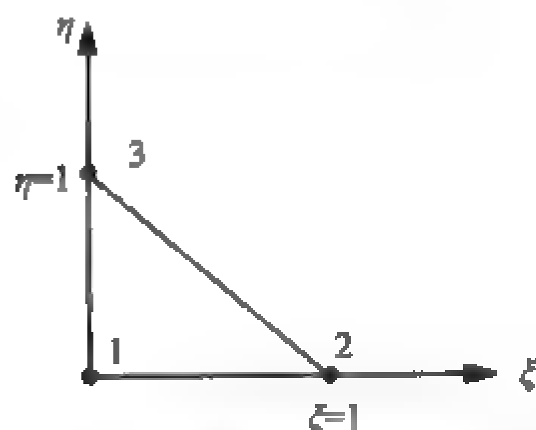


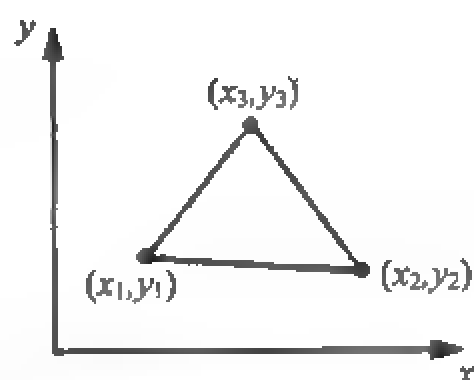
图 4.6 九节点的四边形等参单元

4.3 三角形单元

三角形等参单元的变换如图 4.7 所示.



(a) 自然坐标系中的情况



(b) 自然坐标系到物理坐标系的映射

图 4.7 三角形等参单元

三角形等参单元的映射变换与四边形等参单元相同. 线性三角形单元的形函数为

$$
\begin{aligned}
N_1 &= 1-\xi-\eta \\
N_2 &= \xi \\
N_3 &= \eta
\end{aligned}
\tag{4-27}
$$

而如图 4.8 所示的六节点三角形单元的形函数为

$$
\begin{aligned}
N_1 &= (1-\xi-\eta)(1-2\xi-2\eta) \\
N_2 &= \xi(2\xi-1) \\
N_3 &= \eta(2\eta-1) \\
N_4 &= 4\xi(1-\xi-\eta) \\
N_5 &= 4\xi\eta \\
N_6 &= 4\eta(1-\xi-\eta)
\end{aligned}
\tag{4-28}
$$

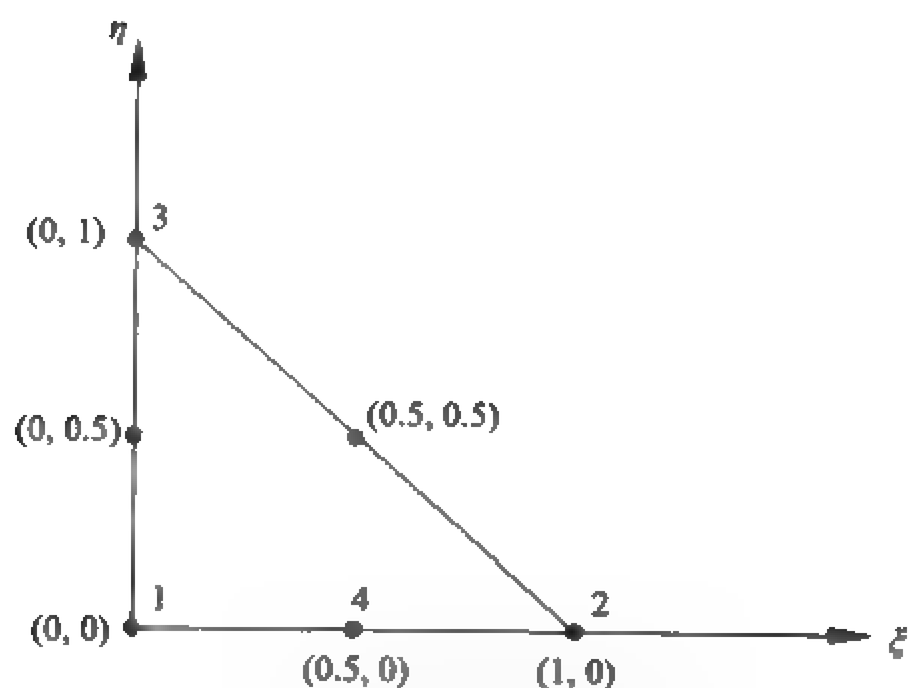


图 4.8　六节点三角形等参单元

4.4 三 维 单 元

三维等参单元的变换如图 4.9 所示.

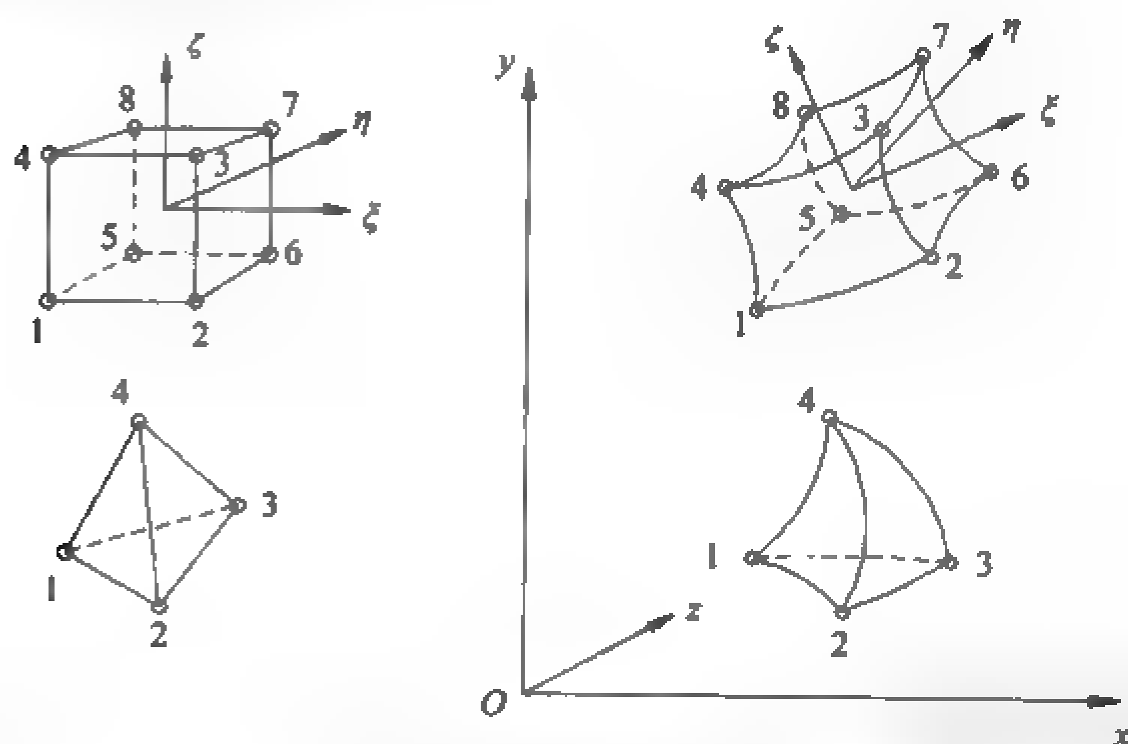


图 4.9　三维等参单元变换

当自然坐标系中单元取为八节点正六面体单元时，单元的形函数定义如下

$$N_1 = \frac{1}{8}(1-\xi)(1-\eta)(1-\zeta)$$

$$N_2 = \frac{1}{8}(1+\xi)(1-\eta)(1-\zeta)$$

$$N_3 = \frac{1}{8}(1+\xi)(1-\eta)(1+\zeta)$$

$$
\begin{aligned}
N_4 &= \frac{1}{8}(1-\xi)(1-\eta)(1+\zeta) \\
N_5 &= \frac{1}{8}(1-\xi)(1+\eta)(1-\zeta) \\
N_6 &= \frac{1}{8}(1+\xi)(1+\eta)(1-\zeta) \\
N_7 &= \frac{1}{8}(1+\xi)(1+\eta)(1+\zeta) \\
N_8 &= \frac{1}{8}(1-\xi)(1+\eta)(1+\zeta)
\end{aligned}
\tag{4-29}
$$

更多节点的单元或其他的等参单元的形函数可参见文献[2].

单元从自然坐标系到物理坐标系映射为

$$
x=\sum_{i=1}^{m}N_i(\xi,\eta,\zeta)x_i\,,\quad y=\sum_{i=1}^{m}N_i(\xi,\eta,\zeta)y_i\,,\quad z=\sum_{i=1}^{m}N_i(\xi,\eta,\zeta)z_i \tag{4-30}
$$

式中，m 为单元形函数的个数．在进行映射变换时，要求单元两个坐标系下的节点编号对应．

单元的节点变量用形函数进行插值，有

$$
u=\sum_{i=1}^{m}N_i(\xi,\eta,\zeta)u_i \tag{4-31}
$$

对应于自然坐标系，形函数的导数为

$$
\begin{aligned}
\frac{\partial N_i(\xi,\eta,\zeta)}{\partial \xi} &= \frac{\partial N_i(\xi,\eta,\zeta)}{\partial x}\frac{\partial x}{\partial \xi}+\frac{\partial N_i(\xi,\eta,\zeta)}{\partial y}\frac{\partial y}{\partial \xi}+\frac{\partial N_i(\xi,\eta,\zeta)}{\partial z}\frac{\partial z}{\partial \xi} \\
\frac{\partial N_i(\xi,\eta,\zeta)}{\partial \eta} &= \frac{\partial N_i(\xi,\eta,\zeta)}{\partial x}\frac{\partial x}{\partial \eta}+\frac{\partial N_i(\xi,\eta,\zeta)}{\partial y}\frac{\partial y}{\partial \eta}+\frac{\partial N_i(\xi,\eta,\zeta)}{\partial z}\frac{\partial z}{\partial \eta} \\
\frac{\partial N_i(\xi,\eta,\zeta)}{\partial \zeta} &= \frac{\partial N_i(\xi,\eta,\zeta)}{\partial x}\frac{\partial x}{\partial \zeta}+\frac{\partial N_i(\xi,\eta,\zeta)}{\partial y}\frac{\partial y}{\partial \zeta}+\frac{\partial N_i(\xi,\eta,\zeta)}{\partial z}\frac{\partial z}{\partial \zeta}
\end{aligned}
\tag{4-32}
$$

写成矩阵形式为

$$
\begin{Bmatrix} \dfrac{\partial}{\partial \xi} \\ \dfrac{\partial}{\partial \eta} \\ \dfrac{\partial}{\partial \zeta} \end{Bmatrix} N_i(\xi,\eta,\zeta)=
\begin{bmatrix}
\dfrac{\partial x}{\partial \xi} & \dfrac{\partial y}{\partial \xi} & \dfrac{\partial z}{\partial \xi} \\
\dfrac{\partial x}{\partial \eta} & \dfrac{\partial y}{\partial \eta} & \dfrac{\partial z}{\partial \eta} \\
\dfrac{\partial x}{\partial \zeta} & \dfrac{\partial y}{\partial \zeta} & \dfrac{\partial z}{\partial \zeta}
\end{bmatrix}
\begin{Bmatrix} \dfrac{\partial}{\partial x} \\ \dfrac{\partial}{\partial y} \\ \dfrac{\partial}{\partial z} \end{Bmatrix} N_i(\xi,\eta,\zeta)=\boldsymbol{J}
\begin{Bmatrix} \dfrac{\partial}{\partial x} \\ \dfrac{\partial}{\partial y} \\ \dfrac{\partial}{\partial z} \end{Bmatrix} N_i(\xi,\eta,\zeta)
\tag{4-33}
$$

式中，$\boldsymbol{J}$ 为 Jacobi 矩阵，且有

$$
\boldsymbol{J}=\frac{\partial(x,y,z)}{\partial(\xi,\eta,\zeta)}=
\begin{bmatrix}
\sum\limits_{i=1}^{m}\dfrac{\partial N_i(\xi,\eta,\zeta)}{\partial \xi}x_i & \sum\limits_{i=1}^{m}\dfrac{\partial N_i(\xi,\eta,\zeta)}{\partial \xi}y_i & \sum\limits_{i=1}^{m}\dfrac{\partial N_i(\xi,\eta,\zeta)}{\partial \xi}z_i \\
\sum\limits_{i=1}^{m}\dfrac{\partial N_i(\xi,\eta,\zeta)}{\partial \eta}x_i & \sum\limits_{i=1}^{m}\dfrac{\partial N_i(\xi,\eta,\zeta)}{\partial \eta}y_i & \sum\limits_{i=1}^{m}\dfrac{\partial N_i(\xi,\eta,\zeta)}{\partial \eta}z_i \\
\sum\limits_{i=1}^{m}\dfrac{\partial N_i(\xi,\eta,\zeta)}{\partial \zeta}x_i & \sum\limits_{i=1}^{m}\dfrac{\partial N_i(\xi,\eta,\zeta)}{\partial \zeta}y_i & \sum\limits_{i=1}^{m}\dfrac{\partial N_i(\xi,\eta,\zeta)}{\partial \zeta}z_i
\end{bmatrix}
\tag{4-34}
$$

对应于物理坐标系，形函数的导数为

$$
\begin{aligned}
\frac{\partial N_i(\xi,\eta,\zeta)}{\partial x} &= \frac{\partial N_i(\xi,\eta,\zeta)}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial N_i(\xi,\eta,\zeta)}{\partial \eta}\frac{\partial \eta}{\partial x} + \frac{\partial N_i(\xi,\eta,\zeta)}{\partial \zeta}\frac{\partial \zeta}{\partial x} \\
\frac{\partial N_i(\xi,\eta,\zeta)}{\partial y} &= \frac{\partial N_i(\xi,\eta,\zeta)}{\partial \xi}\frac{\partial \xi}{\partial y} + \frac{\partial N_i(\xi,\eta,\zeta)}{\partial \eta}\frac{\partial \eta}{\partial y} + \frac{\partial N_i(\xi,\eta,\zeta)}{\partial \zeta}\frac{\partial \zeta}{\partial y} \\
\frac{\partial N_i(\xi,\eta,\zeta)}{\partial z} &= \frac{\partial N_i(\xi,\eta,\zeta)}{\partial \xi}\frac{\partial \xi}{\partial z} + \frac{\partial N_i(\xi,\eta,\zeta)}{\partial \eta}\frac{\partial \eta}{\partial z} + \frac{\partial N_i(\xi,\eta,\zeta)}{\partial \zeta}\frac{\partial \zeta}{\partial z}
\end{aligned} \tag{4-35}
$$

写成矩阵形式为

$$
\begin{Bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{Bmatrix} N_i(\xi,\eta,\zeta) = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} & \frac{\partial \zeta}{\partial x} \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} & \frac{\partial \zeta}{\partial y} \\ \frac{\partial \xi}{\partial z} & \frac{\partial \eta}{\partial z} & \frac{\partial \zeta}{\partial z} \end{bmatrix} \begin{Bmatrix} \frac{\partial}{\partial \xi} \\ \frac{\partial}{\partial \eta} \\ \frac{\partial}{\partial \zeta} \end{Bmatrix} N_i(\xi,\eta,\zeta) = \boldsymbol{R} \begin{Bmatrix} \frac{\partial}{\partial \xi} \\ \frac{\partial}{\partial \eta} \\ \frac{\partial}{\partial \zeta} \end{Bmatrix} N_i(\xi,\eta,\zeta) \tag{4-36}
$$

比较式(4-33)和式(4-36)，可知

$$
\boldsymbol{R} = \boldsymbol{J}^{-1} \tag{4-37}
$$

由 $\mathrm{d}\xi$、$\mathrm{d}\eta$、$\mathrm{d}\zeta$ 在物理坐标系中形成的体积微元为

$$
\mathrm{d}V = \mathrm{d}\boldsymbol{\xi} \cdot (\mathrm{d}\boldsymbol{\eta} \times \mathrm{d}\boldsymbol{\zeta}) \tag{4-38}
$$

而

$$
\begin{aligned}
\mathrm{d}\boldsymbol{\xi} &= \frac{\partial x}{\partial \xi}\mathrm{d}\xi \boldsymbol{i} + \frac{\partial y}{\partial \xi}\mathrm{d}\xi \boldsymbol{j} + \frac{\partial z}{\partial \xi}\mathrm{d}\xi \boldsymbol{k} \\
\mathrm{d}\boldsymbol{\eta} &= \frac{\partial x}{\partial \eta}\mathrm{d}\eta \boldsymbol{i} + \frac{\partial y}{\partial \eta}\mathrm{d}\eta \boldsymbol{j} + \frac{\partial z}{\partial \eta}\mathrm{d}\eta \boldsymbol{k} \\
\mathrm{d}\boldsymbol{k} &= \frac{\partial x}{\partial \zeta}\mathrm{d}\zeta \boldsymbol{i} + \frac{\partial y}{\partial \zeta}\mathrm{d}\zeta \boldsymbol{j} + \frac{\partial z}{\partial \zeta}\mathrm{d}\zeta \boldsymbol{k}
\end{aligned} \tag{4-39}
$$

式中，$\boldsymbol{i}$、$\boldsymbol{j}$、$\boldsymbol{k}$ 是物理坐标系的单元坐标向量．将式(4-39)代入式(4-37)，得

$$
\mathrm{d}V = \begin{vmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{vmatrix} \mathrm{d}\xi\,\mathrm{d}\eta\,\mathrm{d}\zeta = |\boldsymbol{J}|\,\mathrm{d}\xi\,\mathrm{d}\eta\,\mathrm{d}\zeta \tag{4-40}
$$

4.5 等参单元用于弹性力学分析的一般格式

等参单元通常是以位移作为基本变量，进行结构分析有限元的格式对等参单元同样适用，其中的差别只是等参单元的形函数是用自然坐标给出的，等参单元的计算在自然坐标

系中的规则单元内进行，只要将等参单元的变换公式代入有限元的一般格式进行修正即可.

系统的有限元方程为

$$\boldsymbol{M}\ddot{\boldsymbol{a}}(t)+\boldsymbol{C}\dot{\boldsymbol{a}}(t)+\boldsymbol{K}\boldsymbol{a}(t)=\boldsymbol{P}(t) \tag{4-41}$$

式中，$\boldsymbol{M}$ 为系统质量矩阵，$\boldsymbol{M}=\sum \boldsymbol{G}^{\mathrm{T}}\boldsymbol{M}^{\mathrm{e}}\boldsymbol{G}$；$\boldsymbol{C}$ 为系统阻尼矩阵，$\boldsymbol{C}=\sum \boldsymbol{G}^{\mathrm{T}}\boldsymbol{C}^{\mathrm{e}}\boldsymbol{G}$；$\boldsymbol{K}$ 为系统刚度矩阵，$\boldsymbol{K}=\sum \boldsymbol{G}^{\mathrm{T}}\boldsymbol{K}^{\mathrm{e}}\boldsymbol{G}$；$\boldsymbol{P}$ 为系统载荷向量，$\boldsymbol{P}=\sum \boldsymbol{G}^{\mathrm{T}}\boldsymbol{P}^{\mathrm{e}}$．其中，$\boldsymbol{M}^{\mathrm{e}}$、$\boldsymbol{C}^{\mathrm{e}}$、$\boldsymbol{K}^{\mathrm{e}}$ 和 $\boldsymbol{P}^{\mathrm{e}}$ 分别为单元质量矩阵、单元阻尼矩阵、单元刚度矩阵和单元节点载荷向量(其中上标或下标 e 代表单元(element)，用正体)，且有

$$\begin{aligned}
\boldsymbol{M}^{\mathrm{e}} &= \int_{V_{\mathrm{e}}} \rho \boldsymbol{N}^{\mathrm{T}}\boldsymbol{N}\mathrm{d}V \\
\boldsymbol{C}^{\mathrm{e}} &= \int_{V_{\mathrm{e}}} \mu \boldsymbol{N}^{\mathrm{T}}\boldsymbol{N}\mathrm{d}V \\
\boldsymbol{K}^{\mathrm{e}} &= \int_{V_{\mathrm{e}}} \boldsymbol{B}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{B}\mathrm{d}V \\
\boldsymbol{P}^{\mathrm{e}} &= \int_{V_{\mathrm{e}}} \boldsymbol{N}^{\mathrm{T}}\boldsymbol{f}\mathrm{d}V + \int_{S_{\sigma}^{\mathrm{e}}} \boldsymbol{N}^{\mathrm{T}}\bar{\boldsymbol{T}}\mathrm{d}S
\end{aligned} \tag{4-42}$$

利用等参变换计算单元矩阵时只需进行两方面的修改：积分变量(取自然坐标系)和积分限．下面以三维单元为例，对 ξ、η、ζ 坐标系中的立方体单元讨论单元矩阵的计算．

在自然坐标系中有

$$-1\leqslant\xi\leqslant1,\quad -1\leqslant\eta\leqslant1,\quad -1\leqslant\zeta\leqslant1$$

再将体积微元、面积微元和线微元的变换式代入式(4-42)，可得

$$\begin{aligned}
\boldsymbol{M}^{\mathrm{e}} &= \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} \rho \boldsymbol{N}^{\mathrm{T}}\boldsymbol{N}|\boldsymbol{J}|\mathrm{d}\xi\mathrm{d}\eta\mathrm{d}\zeta \\
\boldsymbol{C}^{\mathrm{e}} &= \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} \mu \boldsymbol{N}^{\mathrm{T}}\boldsymbol{N}|\boldsymbol{J}|\mathrm{d}\xi\mathrm{d}\eta\mathrm{d}\zeta \\
\boldsymbol{K}^{\mathrm{e}} &= \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} \boldsymbol{B}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{B}|\boldsymbol{J}|\mathrm{d}\xi\mathrm{d}\eta\mathrm{d}\zeta \\
\boldsymbol{P}^{\mathrm{e}} &= \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} \boldsymbol{N}^{\mathrm{T}}\boldsymbol{f}|\boldsymbol{J}|\mathrm{d}\xi\mathrm{d}\eta\mathrm{d}\zeta + \int_{-1}^{1}\int_{-1}^{1} \boldsymbol{N}^{\mathrm{T}}\bar{\boldsymbol{T}}A\mathrm{d}\eta\mathrm{d}\zeta
\end{aligned} \tag{4-43}$$

($\bar{\boldsymbol{T}}$ 作用在 $\xi=1$ 的面上)

其中

$$|\boldsymbol{J}|=\begin{vmatrix}
\dfrac{\partial x}{\partial \xi} & \dfrac{\partial y}{\partial \xi} & \dfrac{\partial z}{\partial \xi} \\
\dfrac{\partial x}{\partial \eta} & \dfrac{\partial y}{\partial \eta} & \dfrac{\partial z}{\partial \eta} \\
\dfrac{\partial x}{\partial \zeta} & \dfrac{\partial y}{\partial \zeta} & \dfrac{\partial z}{\partial \zeta}
\end{vmatrix} \tag{4-44}$$

$$A=\left[\left(\frac{\partial y}{\partial \eta}\frac{\partial z}{\partial \zeta}-\frac{\partial y}{\partial \zeta}\frac{\partial z}{\partial \eta}\right)^2+\left(\frac{\partial z}{\partial \eta}\frac{\partial x}{\partial \zeta}-\frac{\partial z}{\partial \zeta}\frac{\partial x}{\partial \eta}\right)^2+\left(\frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \zeta}-\frac{\partial x}{\partial \zeta}\frac{\partial y}{\partial \eta}\right)^2\right]^{\frac{1}{2}} \tag{4-45}$$

4.6　数值积分方法

由于单元形状的复杂多变，进行等参变换时 J 与 $|J|$ 都比较复杂．在单元特性矩阵的计算时，尽管采用自然坐标后积分限规则化了，但是除了少数的情况能给出显式积分外，一般都需要进行数值积分．在有限元法中，计算单元特性矩阵常采用 Gauss(高斯)积分．

4.6.1　一维数值积分

对于一个一维问题的数值积分 $\int_a^b F(\xi)\mathrm{d}\xi$，其基本思想是：构造一个多项式 $\phi(\xi)$，使在 ξ_i $(i=1,2,\cdots,n)$ 上有 $\phi(\xi_i)=F(\xi_i)$，然后用近似函数 $\phi(\xi)$ 的积分 $\int_a^b \phi(\xi)\mathrm{d}\xi$ 近似原被积函数 $F(\xi)$ 的积分 $\int_a^b F(\xi)\mathrm{d}\xi$．$\xi_i$ 称为积分点或采样点，其数目和位置决定了函数 $\phi(\xi)$ 对函数 $F(\xi)$ 的近似程度，因而也决定了数值积分的精度．

在 Gauss 积分方法中，积分点 ξ_i 的位置由下述方法确定．首先定义 n 次 Lagrange(拉格朗日)多项式 $P(\xi)$

$$P(\xi)=(\xi-\xi_1)(\xi-\xi_2)\cdots(\xi-\xi_n)=\prod_{j=1}^{n}(\xi-\xi_j) \tag{4-46}$$

再由下列条件确定 n 个积分点的位置

$$\int_a^b \xi^i P(\xi)\mathrm{d}\xi=0 \quad (i=1,2,\cdots,n-1) \tag{4-47}$$

然后构造一个近似多项式 $\phi(\xi)$，使在积分点上有

$$\phi(\xi_i)=F(\xi_i) \quad (i=1,2,\cdots,n) \tag{4-48}$$

该近似函数可由下式确定

$$\phi(\xi)=\sum_{i=1}^{n} l_i^{(n-1)}(\xi)F(\xi_i)+\sum_{i=0}^{n-1}\beta_i\xi^i P(\xi) \tag{4-49}$$

式中，$l_i^{(n-1)}(\xi)$ 是 $n-1$ 阶的 Lagrange 插值多项式

$$l_i^{(n-1)}(\xi)=\frac{(\xi-\xi_1)(\xi-\xi_2)\cdots(\xi-\xi_{i-1})(\xi-\xi_{i+1})\cdots(\xi-\xi_n)}{(\xi_i-\xi_1)(\xi_i-\xi_2)\cdots(\xi_i-\xi_{i-1})(\xi_i-\xi_{i+1})\cdots(\xi_i-\xi_n)} \tag{4-50}$$

对式(4-49)积分，可得

$$\begin{aligned}\int_a^b F(\xi)\mathrm{d}\xi &= \sum_{i=1}^{n}\int_a^b l_i^{(n-1)}(\xi)F(\xi_i)\mathrm{d}\xi+\sum_{i=0}^{n-1}\beta_i\int_a^b \xi^i P(\xi)\mathrm{d}\xi+R \\ &= \sum_{i=1}^{n} H_i F(\xi_i)+R\end{aligned} \tag{4-51}$$

其中

$$H_i=\int_a^b l_i^{(n-1)}(\xi)\mathrm{d}\xi \tag{4-52}$$

式中，H_i 称为积分的权系数，是一个只与积分点的个数和位置有关的参数．

可以看出，Gauss 积分中的积分点是不等间距分布的，近似被积函数 $\phi(\xi)$ 是一个包含

$F(\xi_i)$ $(i=1,2,\cdots,n)$ 和 β_i $(i=0,1,2,\cdots,n-1)$ 共 $2n$ 个系数的 $2n-1$ 次多项式．因此，n 个积分点的 Gauss 积分可达到 $2n-1$ 阶的精度，也就是说如果 $F(\xi)$ 是 $2n-1$ 次多项式，积分将是精确的．

为了便于计算积分点的位置 ξ_i 和权系数 H_i，可把上述公式中的积分限进行规格化，即 $-1\leqslant\xi\leqslant1$．这样计算得到的 ξ_i 和 H_i 对于原积分域 (a,b)，积分点的坐标和积分的权系数分别为

$$\frac{a+b}{2}-\frac{a-b}{2}\xi_i,\quad \frac{b-a}{2}H_i \tag{4-53}$$

4.6.2 二维和三维 Gauss 积分

将一维 Gauss 积分扩展成多重积分，即可得到二维和三维问题的 Gauss 积分．

对于二维问题

$$\begin{aligned} I&=\int_{-1}^{1}\int_{-1}^{1}F(\xi,\eta)\mathrm{d}\xi\mathrm{d}\eta=\int_{-1}^{1}\sum_{j=1}^{n}H_jF(\xi_j,\eta)\mathrm{d}\eta=\sum_{i=1}^{n}H_i\sum_{j=1}^{n}H_jF(\xi_j,\eta_i)\\ &=\sum_{i=1}^{n}\sum_{j=1}^{n}H_iH_jF(\xi_j,\eta_i)=\sum_{i,j=1}^{n}H_{i,j}F(\xi_j,\eta_i) \end{aligned}\tag{4-54}$$

式中，H_i、H_j 就是一维 Gauss 积分的权系数，n 是每个坐标方向的积分点数．

类似地，对于三维数值积分，则有

$$\begin{aligned} I&=\int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1}F(\xi,\eta,\zeta)\mathrm{d}\xi\mathrm{d}\eta\mathrm{d}\zeta=\sum_{m=1}^{n}\sum_{j=1}^{n}\sum_{i=1}^{n}H_iH_jH_mF(\xi_i,\eta_j,\zeta_m)\\ &=\sum_{m,j,i=1}^{n}H_{ijm}F(\xi_i,\eta_j,\zeta_m) \end{aligned}\tag{4-55}$$

在上述的公式中，各个坐标方向上选取的积分点数是相同的，在实际计算中，在 ξ、η 和 ζ 坐标方向上可选取不同的积分点数，即由具体情况采用不同阶的积分方案．表 4.2 中列出了在积分域 $(-1,1)$ 内，$n=1\sim6$ 的 Gauss 积分点位置和权系数的值．

表 4.2 Gauss 积分的积分点坐标和权系数的值

积分点数 n	积分点坐标 ξ_i	积分权系数 H_i
1	0.00000 0.00000 0.00000	2.00000 0.00000 0.00000
2	±0.57735 0.02691 0.89626	1.00000 0.00000 0.00000
3	±0.77459 0.66692 0.41483	0.55555 0.55555 0.55556
	0.00000 0.00000 0.00000	0.88888 0.88888 0.88889
4	±0.86113 0.63115 0.94053	0.34785 0.48451 0.37454
	±0.33998 0.10435 0.84856	0.65214 0.51548 0.62546
5	±0.90617 0.98459 0.38664	0.23692 0.68850 0.56189
	±0.53846 0.93101 0.05683	0.47862 0.86704 0.99366
	0.00000 0.00000 0.00000	0.56888 0.88888 0.88889

续表

积分点数 n	积分点坐标 ξ_i	积分权系数 H_i
6	±0.93246 95142 03152	0.17132 44923 79170
	±0.66120 93864 66265	0.36076 15730 48139
	±0.23861 91860 83197	0.46791 39345 72691

在等参单元的数值积分计算中，还需要确定数值积分的阶次．数值积分阶次的选择要根据保证积分的精度和使结构总体刚度矩阵非奇异，具体可参见文献[2]．

4.7 应用问题及 MATLAB 程序

【例 4.1】 用 Gauss 积分计算下列一维、二维、三维函数的数值积分：

(1) $f(x)=1+x^2-3x^3+4x^5 \quad (-1\leqslant x\leqslant 1)$；

(2) $f(x,y)=1+4xy-3x^2y^2+x^4y^6 \quad (-1\leqslant x\leqslant 1,-1\leqslant y\leqslant 1)$；

(3) $f(x,y,z)=1+4x^2y^2-3x^2z^4+y^4z^6$；

用 MATLAB 编程计算，积分值为

(1) 2.6667；

(2) 2.7810；

(3) 10.1841．

其中用到的函数为 GaussPoint1(No_points)，用于计算 Gauss 积分的积分点和权系数．

M 文件如下：

```
%-----------------------------------------------------------------------
% Example 4.1
%-----------------------------------------------------------------------
%   Gauss quadrature of a function in one, two, three dimension
% Problem descriptions
%   Integrate:
%   (1)  f(x)=1+x^2-3*x^3+4*x^5  (-1<x<1)
%   (2)  f(x,y)=1+4*x*y-3*x^2*y^2+x^4*y^6   (-1<x<1, -1<y<1)
%   (3)  f(x,y,z)=1+4*x^2*y^2-3*x^2*z^4+y^4*z^6  (-1<(x,y,z)<1)
% Variable descriptions
%   points = integration (or sampling) points
%   weights = weighting coefficients
%   No_INTpoint_x = number of integration points along x-axis
%   No_INTpoint_t = number of integration points along y-axis
%   No_INTpoint_z = number of integration points along z-axis
%-----------------------------------------------------------------------
%-------------------------------
%  (1) Select the problem
%-------------------------------
clear; clc;
```

```
Opt_problem=1;                          % option for type of the beam:
                                        % =1 One-dimensional function
                                        % =2 Two-dimensional function
                                        % =3 Three-dimensional function
No_INTpoint_x=5;                        % No_INTpoint_x >= 0.5*(Order_x+1)
No_INTpoint_y=0;                        % No_INTpoint_y >= 0.5*(Order_y+1)
No_INTpoint_z=0;                        % No_INTpoint_z >= 0.5*(Order_z+1)
%-------------------------------
%  (2) initialize the vectors
%-------------------------------
   pointx=zeros(No_INTpoint_x,1);
   weightx=zeros(No_INTpoint_x,1);

   pointy=zeros(No_INTpoint_y,1);
   weighty=zeros(No_INTpoint_y,1);

   pointz=zeros(No_INTpoint_z,1);
   weightz=zeros(No_INTpoint_z,1);
%--------------------------------------------
%  (3) compute numerical integration
%--------------------------------------------
value=0.0;

switch Opt_problem
  case 1
    [pointx,weightx]=GaussPoint1(No_INTpoint_x);
                              % extract integration points and weights
    for ii=1:No_INTpoint_x
      x=pointx(ii);                     % sampling points in x-axis
      wtx=weightx(ii);                  % weighting coefficients in x-axis
      func=1+x^2-3*x^3+4*x^5;                        % evaluate function
      value=value+func*wtx;
    end
  case 2
    [pointx,weightx]=GaussPoint1(No_INTpoint_x);
    [pointy,weighty]=GaussPoint1(No_INTpoint_y);
                              % extract integration points and weights
    for ii=1:No_INTpoint_x
      x=pointx(ii);                     % sampling points in x-axis
      wtx=weightx(ii);                  % weighting coefficients in x-axis
      for ij=1:No_INTpoint_y
        y=pointy(ij);                   % sampling points in y-axis
        wty=weighty(ij) ;               % weighting coefficients in y-axis
        func=1+4*x*y-3*x^2*y^2+x^4*y^6;              % evaluate function
        value=value+func*wtx*wty;
      end
```

```
    end
  case 3
    [pointx,weightx]=GaussPoint1(No_INTpoint_x);
    [pointy,weighty]=GaussPoint1(No_INTpoint_y);
    [pointz,weightz]=GaussPoint1(No_INTpoint_z);
                                        % extract integration points and weights
    for ii=1:No_INTpoint_x
      x=pointx(ii);                     % sampling points in x-axis
      wtx=weightx(ii);                  % weighting coefficients in x-axis
      for ij=1:No_INTpoint_y
        y=pointy(ij);                   % sampling points in y-axis
        wty=weighty(ij) ;               % weighting coefficients in y-axis
        for ik=1:No_INTpoint_z
          z=pointz(ik);                 % sampling points in z-axis
          wtz=weightz(ik) ;             % weighting coefficients in z-axis
          func=1+4*x^2*y^2-3*x^2*z^4+y^4*z^6;        % evaluate function
          value=value+func*wtx*wty*wtz;
        end
      end
    end
  otherwise
    disp('wrong number for problem')
end

value                                   % print the solution
%---------------------------------------------------------------------
%   The end
%---------------------------------------------------------------------


%---------------------------------------------------------------------
function [point1,weight1]=GaussPoint1(No_points)
%---------------------------------------------------------------------
%  Purpose:
%     determine the integration points and weighting coefficients of
%      Gauss quadrature
%  Variable Description:
%     No_points - number of integration points
%     point1  - vector containing integration points
%     weight1 - vector containing weighting coefficients
%-----------------------------------------------------------------
%-----------------------------------------
%  (1) initialization of the vectors
%-----------------------------------------
  points=zeros(No_points,1);
  weights=zeros(No points,1);
%----------------------------------------------------
%  (2) find corresponding integration points and weights
```

```
%-------------------------------------------------------------------
if No points==1                  % 1-point quadrature rule
   point1(1) 0.0;
   weight1(1)=2.0;
elseif No_points==2              % 2-point quadrature rule
   point1(1)=-0.577350269189626;
   point1(2)=-point1(1);

   weight1(1)=1.0;
   weight1(2)=weight1(1);
elseif No points==3              % 3-point quadrature rule
   point1(1)=-0.774596669241483;
   point1(2)=0.0;
   point1(3)=-point1(1);

   weight1(1)=0.555555555555556;
   weight1(2)=0.888888888888889;
   weight1(3)=weight1(1);
elseif No_points==4              % 4-point quadrature rule
   point1(1)=-0.861136311594053;
   point1(2)=-0.339981043584856;
   point1(3)=-point1(2);
   point1(4)=-point1(1);

   weight1(1)=0.347854845137454;
   weight1(2)=0.652145154862546;
   weight1(3)=weight1(2);
   weight1(4)=weight1(1);
elseif No_points==5              % 5-point quadrature rule
   point1(1)=-0.906179845938664;
   point1(2)=-0.538469310105683;
   point1(3)=0.0;
   point1(4)=-point1(2);
   point1(5)=-point1(1);

   weight1(1)=0.236926885056189;
   weight1(2)=0.478628670499366;
   weight1(3)=0.568888888888889;
   weight1(4)=weight1(2);
   weights(5)=weight1(1);
elseif  No_points==6             % 6-point quadrature rule
   point1(1)=-0.932469514203152;
   point1(2)=-0.661209386466265;
   point1(3)=-0.238619186083197;
   point1(4)=-point1(3);
   point1(5)=-point1(2);
   point1(6)=-point1(1);
```

```
    weight1(1)=0.171324492379170;
    weight1(2)=0.360761573048139;
    weight1(3)=0.467913934572691;
    weight1(4)=weight1(3);
    weight1(5)=weight1(2);
    weight1(6)=weight1(1);
else
   disp('number of integration points should be')
   disp('> 0 or < 7')
end
%------------------------------------------------------------------
%    The end
%------------------------------------------------------------------
```

【例 4.2】 用二维等参单元对处于平面应力状态的短悬臂梁(见图 4.10)进行静力分析。梁的长度和高度分别为 8m、1m，梁材料的弹性模量为 $E=1.0\times10^6$ Pa，泊松比为 0.3，梁端部的集中载荷为 1000N。

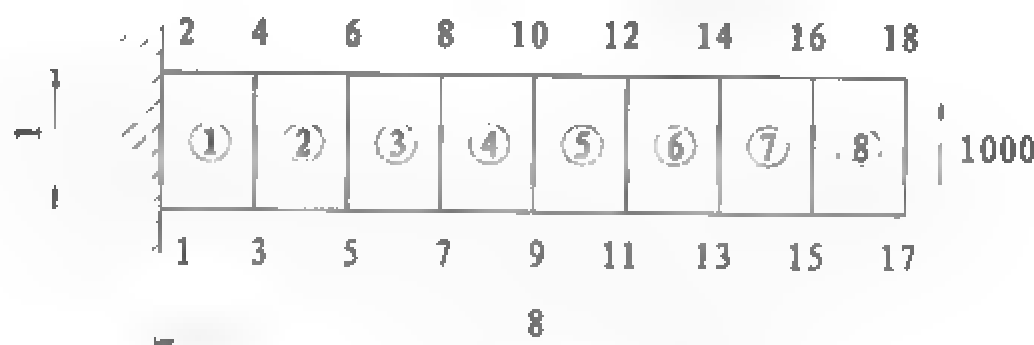


图 4.10　悬臂梁结构

用四节点四边形等参单元计算该短悬臂梁问题，单元的划分如图 4.10 所示。

用 MATLAB 编程计算，其中用到的函数有：

- GaussPoint2(No_points) ——用于计算 Gauss 积分的积分点和权系数。
- Materialiso(iopt, E, u)——用于计算各向同性材料的弹性矩阵。
- FemIsoq4(rvalue, svalue)——用于计算四节点四边形等参单元的形函数及其在自然坐标系下的导数。
- FemJacobi2(No_nodeEl, dhdr, dhds, xcoord, ycoord)——用于计算 Jacobi 矩阵。
- FemDeriv2(No_nodeEl, dhdr, dhds, invjacob)——用于计算等参单元的形函数在物理坐标系下的导数。
- FemKine2D(No_nodeEl, dhdx, dhdy)——用于确定应变与位移的关系。
- FemEldof(nd, No_nodeEl, No_dof)——用于确定单元在系统中的自由度。
- FemAsmbl1(kk, k, index)——用于组集系统矩阵。
- FemAplyc2(kk, ff, bcdof, bcval)——用于对系统方程施加约束条件。

M 文件如下：

```
%--------------------------------------------------------------------------
% Example 4.2
%--------------------------------------------------------------------------
%   plane stress analysis of a cantilever beam using isoparametric
%   four-node elements. (see Fig. 4-10 for the finite element mesh)
%--------------------------------------------------------------------------
%------------------------------------
% (0) input data for control parameters
%------------------------------------
clear; clc;

No_element=8;                     % number of elements
No_nodeEl=4;                      % number of nodes per element
No_dof=2;                         % number of dofs per node
No_nodeSys=18;                    % total number of nodes in system
Sys_dof=No_nodeSys*No_dof;        % total system dofs
El_dof=No_nodeEl*No_dof;          % degrees of freedom per element
emodule=1e6;                      % elastic modulus
poisson=0.3;                      % Poisson's ratio
nglx=2; ngly=2;                   % 2x2 Gauss quadrature
nglxy=nglx*ngly;                  % number of sampling points per element
%---------------------------------
% (1) input nodal coordinate values
%---------------------------------
      % x, y coordinates of nodes in global coordinate system
gcoord=[ 0.0  0.0;   0.0  1.0;
        0.5  0.0;   0.5  1.0;
        1.0  0.0;   1.0  1.0;
        1.5  0.0;   1.5  1.0;
        2.0  0.0;   2.0  1.0;
        2.5  0.0;   2.5  1.0;
        3.0  0.0;   3.0  1.0;
        3.5  0.0;   3.5  1.0;
        4.0  0.0;   4.0  1.0];
%---------------------------------------------------
% (2) input data for nodal connectivity for each element
%---------------------------------------------------
%      element     node code
%        No.    i    j    k    m
nodes=[   1    1    3    4    2;
        2    3    5    6    4;
        3    5    7    8    6;
        4    7    9   10    8;
        5    9   11   12   10;
        6   11   13   14   12;
        7   13   15   16   14;
        8   15   17   18   16];
```

```
nodes=[nodes(:,2:5)];
%-----------------------------------
%  (3) input data for boundary conditions
%-----------------------------------
bcdof=[1  2  3  4];        % first four dofs are constrained
bcval=[0  0  0  0];        % whose described values are 0
%-----------------------------------
%  (4) initialization of matrices and vectors
%-----------------------------------
ff=zeros(Sys_dof,1);          % system force vector
kk=zeros(Sys_dof,Sys_dof);    % system matrix
displmt=zeros(Sys_dof,1);     % system displacement vector
eldisp=zeros(El_dof,1);       % element displacement vector
stress=zeros(nglxy,3);        % matrix containing stress components
strain=zeros(nglxy,3);        % matrix containing strain components
index=zeros(El_dof,1);        % index vector
kinmtx2=zeros(3,El_dof);      % kinematic matrix
matmtrx=zeros(3,3);           % constitutive matrix
%-----------------
%  (5) force vector
%-----------------
ff(34)=500;                 % force applied at node 17 in y-axis
ff(36)=500;                 % force applied at node 18 in y-axis
%-------------------------------------------------------------
%  (6) computation of element matrices and vectors and their assembly
%-------------------------------------------------------------
[point2,weight2]=GaussPoint2(nglx,ngly);   % sampling points & weights
matmtrx=Materialiso(1,emodule,poisson);    % compute constitutive matrix

for iel=1:No_element        % loop for the total number of elements

for i=1:No_nodeEl
nd(i)=nodes(iel,i);         % extract connected node for (iel)-th element
xcoord(i)=gcoord(nd(i),1);     % extract x value of the node
ycoord(i)=gcoord(nd(i),2);     % extract y value of the node
end

k=zeros(El_dof,El_dof);      % initialization of element matrix to zero
%-----------------------------------------------
%  (7) numerical integration and compute element matrix
%-----------------------------------------------
for intx=1:nglx
x=point2(intx,1);                    % sampling point in x-axis
wtx=weight2(intx,1);                 % weight in x-axis
for inty=1:ngly
y=point2(inty,2);                    % sampling point in y-axis
wty=weight2(inty,2) ;                % weight in y-axis
```

```
[shape,dhdr,dhds]=FemIsoq4(x,y);       % compute shape functions and
                                       % derivatives at sampling point
jacob2=FemJacobi2(No_nodeEl,dhdr,dhds,xcoord,ycoord);  % compute Jacobian

detjacob=det(jacob2);                  % determinant of Jacobian
invjacob=inv(jacob2);                  % inverse of Jacobian matrix

[dhdx,dhdy]=FemDeriv2(No_nodeEl,dhdr,dhds,invjacob);  % derivatives w.r.t.
                                                       % physical coordinate
kinmtx2=FemKine2D(No_nodeEl,dhdx,dhdy);       % compute kinematic matrix

k=k+kinmtx2'*matmtrx*kinmtx2*wtx*wty*detjacob;     % compute element matrix

end
end                                    % end of numerical integration loop

index=FemEldof(nd,No_nodeEl,No_dof);
% extract system dofs associated with element

kk=FemAsmbl1(kk,k,index);              % assemble element matrices

end
%  -------------------------
%  (8) apply boundary conditions
%--------------------------
[kk,ff]=FemAplyc2(kk,ff,bcdof,bcval);
%---------------------------
%  (9) solve the matrix equation
%---------------------------
displmt=kk\ff;

num=1:1:Sys_dof;
displacement=[num' displmt]            % print nodal displacements
%-------------------------------
%  (10) element stress computation
%-------------------------------
for ielp=1:No_element     % loop for the total number of elements

for i=1:No_nodeEl
nd(i)=nodes(ielp,i);      % extract connected node for (iel)-th element
xcoord(i)=gcoord(nd(i),1);            % extract x value of the node
ycoord(i)=gcoord(nd(i),2);            % extract y value of the node
end
%---------------------------
%  (10.1) numerical integration
%   ------------------------
```

```
intp=0;
for intx=1:nglx
x=point2(intx,1);                    % sampling point in x-axis
wtx=weight2(intx,1);                 % weight in x-axis
for inty=1:ngly
y=point2(inty,2);                    % sampling point in y-axis
wty=weight2(inty,2) ;                % weight in y-axis
intp=intp+1;

[shape,dhdr,dhds]=FemIsoq4(x,y);     % compute shape functions and
                                     % derivatives at sampling point
jacob2=FemJacobi2(No_nodeEl,dhdr,dhds,xcoord,ycoord);  % compute Jacobian

detjacob=det(jacob2);                % determinant of Jacobian
invjacob=inv(jacob2);                % inverse of Jacobian matrix

[dhdx,dhdy]=FemDeriv2(No_nodeEl,dhdr,dhds,invjacob);  % derivatives w.r.t.
                                           % physical coordinate
kinmtx2=FemKine2D(No_nodeEl,dhdx,dhdy);          % kinematic matrix

index=FemEldof(nd,No_nodeEl,No_dof);  % extract system dofs for the element
%-----------------------------------------
%  (10.2) extract element displacement vector
%-----------------------------------------
for i=1:El_dof
eldisp(i)=displmt(index(i));
end

kinmtx2=FemKine2D(No nodeEl,dhdx,dhdy);    % compute kinematic matrix

estrain=kinmtx2*eldisp;              % compute strains
estress=matmtrx*estrain;             % compute stresses

for i=1:3
strain(intp,i)=estrain(i);           % store for each element
stress(intp,i)=estress(i);           % store for each element
end

location=[ielp,intx,inty]            % print location for stress
stress(intp,:)                       % print stress values

end
end                                  % end of integration loop

end
%-------------------------------------------------------------------
%    The end
```

```
%----------------------------------------------------------------------

%----------------------------------------------------------------------
function [point2,weight2]=GaussPoint2(No_point_x,No_point_y)
%----------------------------------------------------------------------
%  Purpose:
%    determine the integration points and weighting coefficients
%    of Gauss quadrature for two-dimensional integration
%----------------------------------------------------------------------
%-------------------------------------------------------------
%  (1) determine the largest one between No_point_x and No_point_y
%-------------------------------------------------------------
  if No_point_x > No_point_y
    ng=No_point_x;
  else
    ng=No_point_y;
  end
%----------------------------------
%  (2) initialization of coefficient vector
%----------------------------------
  point2=zeros(ng,2);
  weight2=zeros(ng,2);
%----------------------------------------------
%  (3) find corresponding integration points and weights
%----------------------------------------------
 [pointx,weightx]=GaussPoint1(No_point_x);  % quadrature rule for x-axis
 [pointy,weighty]=GaussPoint1(No_point_y);  % quadrature rule for y-axis
%-------------------------------
%  (4) quadrature for two-dimension
%-------------------------------
 for ii=1:No_point_x                        % quadrature in x-axis
  point2(ii,1)=pointx(ii);
  weight2(ii,1)=weightx(ii);
 end
 for ij=1:No_point_y                        % quadrature in y-axis
  point2(ij,2)=pointy(ij);
  weight2(ij,2)=weighty(ij);
 end
%----------------------------------------------------------------------
%   The end
%----------------------------------------------------------------------

%----------------------------------------------------------------------
function [matmtrx]=Materialiso(iopt,E,u)
%----------------------------------------------------------------------
%  Purpose:
%    determine the constitutive equation for isotropic material
```

```
% Variable Description:
%    E - elastic modulus
%    u - Poisson's ratio
%    iopt=1 - plane stress analysis
%    iopt=2 - plane strain analysis
%    iopt=3 - axisymmetric analysis
%    iopt=4 - three dimensional analysis
%-------------------------------------------------------------------------
 if iopt==1        % plane stress
   matmtrx= E/(1-u*u)* ...
   [1  u   0; ...
   u   1   0; ...
   0   0  (1-u)/2];
 elseif iopt==2     % plane strain
   matmtrx= E/((1+u)*(1-2*u))* ...
   [(1-u)  u      0;
   u    (1-u)     0;
   0      0   (1-2*u)/2];
 elseif iopt==3     % axisymmetry
   matmtrx= E/((1+u)*(1-2*u))* ...
   [(1-u)  u     u     0;
   u    (1-u)    u     0;
   u      u   (1-u)    0;
   0      0     0   (1-2*u)/2];
 else     % three-dimension
   matmtrx= E/((1+u)*(1-2*u))* ...
   [(1-u)  u      u        0         0          0;
   u     (1-u)    u        0         0          0;
   u        u    (1-u)     0         0          0;
   0        0      0    (1-2*u)/2   0          0;
   0        0      0        0      (1-2*u)/2    0;
   0        0      0        0         0       (1-2*u)/2];
 end
%-------------------------------------------------------------------------
%   The end
%-------------------------------------------------------------------------

%-------------------------------------------------------------------------
function [shapeq4,dhdrq4,dhdsq4]=FemIsoq4(rvalue,svalue)
%-------------------------------------------------------------------------
% Purpose:
%    compute isoparametric four-node quadrilateral shape functions and
     their derivatives
%    at the selected (integration) point in terms of the natural coordinate
% Variable Description:
%    shapeq4 - shape functions for four-node element
%    dhdrq4 - derivatives of the shape functions
```

```
%    dhdsq4 - derivatives of the shape functions
%    rvalue - r coordinate value of the selected point
%    svalue - s coordinate value of the selected point
% Notes:
%    1st node at (-1,-1), 2nd node at (1,-1), 3rd node at (1,1), 4th
%    node at (-1,1)
%-------------------------------------------------------------------------
%----------------
% (1) shape functions
%--------------------
 shapeq4(1)=0.25*(1-rvalue)*(1-svalue);
 shapeq4(2)=0.25*(1+rvalue)*(1-svalue);
 shapeq4(3)=0.25*(1+rvalue)*(1+svalue);
 shapeq4(4)=0.25*(1-rvalue)*(1+svalue);
%----------------
% (2) derivatives
%----------------
 dhdrq4(1)=-0.25*(1-svalue);
 dhdrq4(2)=0.25*(1-svalue);
 dhdrq4(3)=0.25*(1+svalue);
 dhdrq4(4)=-0.25*(1+svalue);

 dhdsq4(1)=-0.25*(1-rvalue);
 dhdsq4(2)=-0.25*(1+rvalue);
 dhdsq4(3)=0.25*(1+rvalue);
 dhdsq4(4)=0.25*(1-rvalue);
%-------------------------------------------------------------------------
%   The end
%-------------------------------------------------------------------------


%-------------------------------------------------------------------------
function [jacob2]=FemJacobi2(No_nodeEl,dhdr,dhds,xcoord,ycoord)
%-------------------------------------------------------------------------
% Purpose:
%    determine the Jacobian for two-dimensional mapping
% Variable Description:
%    jacob2 - Jacobian for one-dimension
%    No_nodeEl - number of nodes per element
%    dhdr - derivative of shape functions w.r.t. natural coordinate r
%    dhds - derivative of shape functions w.r.t. natural coordinate s
%    xcoord - x axis coordinate values of nodes
%    ycoord - y axis coordinate values of nodes
%-------------------------------------------------------------------------
 jacob2=zeros(2,2);

 for i=1: No_nodeEl
 jacob2(1,1)=jacob2(1,1)+dhdr(i)*xcoord(i);
```

```
 jacob2(1,2)=jacob2(1,2)+dhdr(i)*ycoord(i);
 jacob2(2,1)=jacob2(2,1)+dhds(i)*xcoord(i);
 jacob2(2,2)=jacob2(2,2)+dhds(i)*ycoord(i);
 end
%-------------------------------------------------------------------
%   The end
%-------------------------------------------------------------------


%-------------------------------------------------------------------
function [dhdx,dhdy]=FemDeriv2(No_nodeEl,dhdr,dhds,invjacob)
%-------------------------------------------------------------------
%  Purpose:
%     determine derivatives of 2-D isoparametric shape functions with
%     respect to physical coordinate system
%  Variable Description:
%     dhdx - derivative of shape function w.r.t. physical coordinate x
%     dhdy - derivative of shape function w.r.t. physical coordinate y
%     No_nodeEl - number of nodes per element
%     dhdr - derivative of shape functions w.r.t. natural coordinate r
%     dhds - derivative of shape functions w.r.t. natural coordinate s
%     invjacob - inverse of 2-D Jacobian matrix
%-------------------------------------------------------------------
 for i=1: No_nodeEl
 dhdx(i)=invjacob(1,1)*dhdr(i)+invjacob(1,2)*dhds(i);
 dhdy(i)=invjacob(2,1)*dhdr(i)+invjacob(2,2)*dhds(i);
 end
%-------------------------------------------------------------------
%   The end
%-------------------------------------------------------------------


%-------------------------------------------------------------------
function [kinmtx2]=FemKine2D(No_nodeEl,dhdx,dhdy)
%-------------------------------------------------------------------
%  Purpose:
%     determine the kinematic equation between strains and displacements
%     for two-dimensional solids
%  Variable Description:
%     No_nodeEl - number of nodes per element
%     dhdx - derivatives of shape functions with respect to x
%     dhdy - derivatives of shape functions with respect to y
%-------------------------------------------------------------------

 for i=1: No_nodeEl
 i1=(i-1)*2+1;
 i2=i1+1;
 kinmtx2(1,i1)=dhdx(i);
 kinmtx2(2,i2)=dhdy(i);
```

```
 kinmtx2(3,i1)=dhdy(i);
 kinmtx2(3,i2)=dhdx(i);
 end
%------------------------------------------------------------------------
%   The end
%------------------------------------------------------------------------

%------------------------------------------------------------------------
function [index]=FemEldof(nd, No_nodeEl,No_dof)
%------------------------------------------------------------------------
%  Purpose:
%     Compute system dofs associated with each element
%  Variable Description:
%     index - system dof vector associated with element "iel"
%     nd - node connectivity for the (iel)-th element
%     No nodeEl - number of nodes per element
%     No_dof - number of dofs per node
%------------------------------------------------------------------------
   k=0;
   for i=1: No_nodeEl
     start = (nd(i)-1)* No_dof;
       for j=1:No_dof
         k=k+1;
         index(k)=start+j;
       end
   end
%------------------------------------------------------------------------
%   The end
%------------------------------------------------------------------------

%------------------------------------------------------------------------
function [kk]=FemAsmbl1(kk,k,index)
%------------------------------------------------------------------------
%  Purpose:
%     Assembly of element matrices into the system matrix
%  Variable Description:
%     kk - system matrix
%     k  - element matrix
%     index - d.o.f. vector associated with an element
%------------------------------------------------------------------------
 edof = length(index);
 for i=1:edof
   ii=index(i);
     for j=1:edof
       jj=index(j);
         kk(ii,jj)=kk(ii,jj)+k(i,j);
     end
```

```
 end
%------------------------------------------------------------------------
%   The end
%------------------------------------------------------------------------

%------------------------------------------------------------------------
function [kk,ff]=FemAplyc2(kk,ff,bcdof,bcval)
%------------------------------------------------------------------------
%  Purpose:
%     Apply constraints to matrix equation [kk]{x}={ff}
%  Variable Description:
%     kk - system matrix before applying constraints
%     ff - system vector before applying constraints
%     bcdof - a vector containing constrained d.o.f
%     bcval - a vector containing contained value
%------------------------------------------------------------------------
 n=length(bcdof);
 sdof=size(kk);

 for i=1:n
    c=bcdof(i);
    for j=1:sdof
       kk(c,j)=0;
    end
    kk(c,c)=1;
    ff(c)=bcval(i);
 end
%------------------------------------------------------------------------
%   The end
%------------------------------------------------------------------------
```

计算结果：梁自由端部沿 y 方向的位移为 0.2238m，梁固定端附近积分点处的弯曲应力为 11 945Pa.

第 5 章　梁与刚架结构

5.1　基本单元分析

5.1.1　Euler-Bernoulli 梁单元

对梁结构的分析，可以用具有两个端部节点的一维单元进行，单元的变形为横向位移 v 和转角 θ，如图 5.1 所示，称为梁单元(beam element).

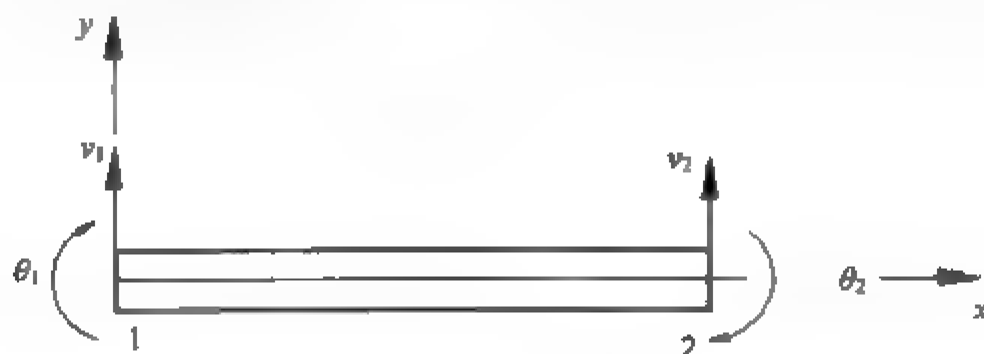


图 5.1　梁单元

假设不考虑梁的剪切变形，可以采用 Euler-Bernoulli (欧拉-伯努利)梁的假设：变形前垂直于梁中心线的截面在变形后仍保持垂直于梁的中心线，如图 5.2 所示. 按照这一假设，梁弯曲变形时的运动方程为

$$\rho A\frac{\partial^2 v}{\partial t^2}+\frac{\partial^2}{\partial x^2}(EI\frac{\partial^2 v}{\partial x^2})=q(x,t) \tag{5-1}$$

式中，$v(x,t)$ 为梁的横向位移；ρ 为梁的质量密度；A 为梁的截面面积；E 为材料的弹性模量；I 为梁的惯性矩；$q(x,t)$ 为外载荷；t 和 x 分别为时间和梁沿中心轴的坐标.

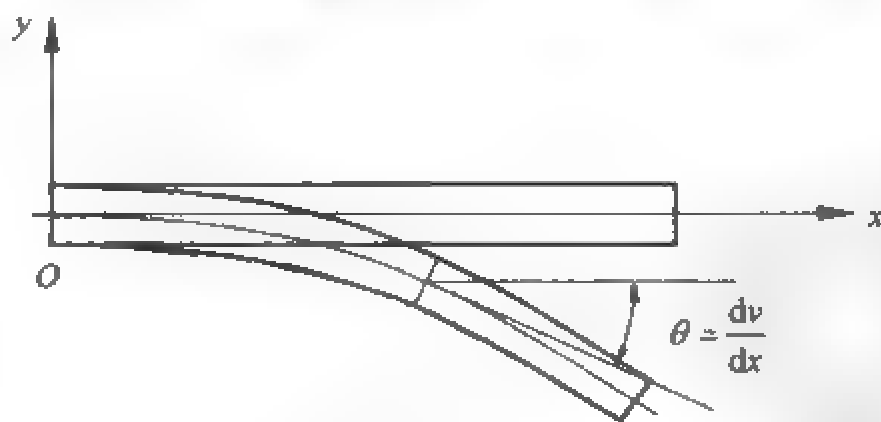


图 5.2　Euler-Bernoulli 梁

利用加权余量法(weighted residual method)[1]，式(5-1)的残差为

$$I=\int_0^L\left(\rho A\frac{\partial^2 v}{\partial t^2}+\frac{\partial^2}{\partial x^2}(EI\frac{\partial^2 v}{\partial x^2})-q\right)w\mathrm{d}x=0 \tag{5-2}$$

式中，L 为梁的长度，w 为试探函数.

对式(5-2)的第二项进行分部积分，有

$$I=\left[\int_0^L \rho A\frac{\partial^2 v}{\partial t^2}w\mathrm{d}x+\int_0^L EI\frac{\partial^2 v}{\partial x^2}\frac{\partial^2 w}{\partial x^2}\mathrm{d}x-\int_0^L \rho w\mathrm{d}x\right]-\left[Vw-M\frac{\partial w}{\partial x}\right]_0^L=0 \tag{5-3}$$

式中：$V=EI(\partial^3 v/\partial x^3)$，为梁的剪力；$M=EI(\partial^2 v/\partial x^2)$，为梁的弯矩.

在有限元法中，将梁离散化后，上述方程

$$I=\sum_{i=1}^{n}\left[\int_{\Omega^e} \rho A\frac{\partial^2 v}{\partial t^2}w\mathrm{d}x+\int_{\Omega^e} EI\frac{\partial^2 v}{\partial x^2}\frac{\partial^2 w}{\partial x^2}\mathrm{d}x-\int_{\Omega^e} qw\mathrm{d}x\right]-\left[Vw-M\frac{\partial w}{\partial x}\right]_0^L=0 \tag{5-4}$$

式中，Ω^e 为单元的定义域，n 为单元的数目.

根据有限元的方法，单元的位移模式应满足完备性和协调性[1]. 如图 5.1 所示梁单元的横向变形和转角是连续的，在节点处须有表示横向位移和转角的变量 v_i 和 θ_i，因而每个单元具有 4 个变量，并且由 Euler-Bernoulli 假设有 $\theta=\mathrm{d}v/\mathrm{d}x$，取单元的位移模式为三阶多项式函数

$$v(x)=c_0+c_1x+c_2x^2+c_3x^3 \tag{5-5}$$

则转角为

$$\theta(x)=c_1+2c_2x+3c_3x^2 \tag{5-6}$$

单元的节点位移可表示为

$$\boldsymbol{\delta}^e=\begin{bmatrix}v_1 & \theta_1 & v_2 & \theta_2\end{bmatrix}^{\mathrm{T}} \tag{5-7}$$

将单元节点的坐标代入，有

$$\left.\begin{aligned} v(0)=c_0=v_1\\ \theta(0)=c_1=\theta_1\\ v(l)=c_0+c_1l+c_2l^2+c_3l^3=v_2\\ \theta(l)=c_1+2c_2l+3c_3l^2=\theta_2\end{aligned}\right\} \tag{5-8}$$

式(5-8)写成矩阵形式为

$$\boldsymbol{\delta}^e=\begin{Bmatrix}v_1\\ \theta_1\\ v_2\\ \theta_2\end{Bmatrix}=\begin{bmatrix}1&0&0&0\\0&1&0&0\\1&l&l^2&l^3\\0&1&2l&3l^2\end{bmatrix}\begin{Bmatrix}c_0\\c_1\\c_2\\c_3\end{Bmatrix}=\boldsymbol{AC} \tag{5-9}$$

其中

$$\boldsymbol{A}=\begin{bmatrix}1&0&0&0\\0&1&0&0\\1&l&l^2&l^3\\0&1&2l&3l^2\end{bmatrix},\qquad \boldsymbol{C}=\begin{Bmatrix}c_0\\c_1\\c_2\\c_3\end{Bmatrix}$$

求解式(5-9)，可得到由节点变量 v_i 和 θ_i 表示的待定系数 c_i. 将这些系数 c_i 再代入式(5-5)，得

$$v(x)=N_1(x)v_1+N_2(x)\theta_1+N_3(x)v_2+N_4(x)\theta_2=\boldsymbol{N}\boldsymbol{\delta}^e \tag{5-10}$$

其中

$$N_1=1-3\xi^2+2\xi^3,\quad N_2(x)=(\xi-2\xi^2+\xi^3)l$$

$$N_3(x)=3\xi^2-2\xi^3,\quad N_4(x)=(-\xi^2+\xi^3)l$$

$$\xi=\frac{x-x_1}{l},\quad N=\begin{bmatrix}N_1 & N_2 & N_3 & N_4\end{bmatrix}$$

式(5-10)中的函数 $N_i(x)$ 称为 Hermite 形函数，可保证相邻单元在边界上的横向位移和转角的连续性.

将上述形函数代入式(5-4)中，并且由 Galerkin 法在方程中取试探函数为 $w_i=N_i$，则有

$$\int_{\Omega^e}\rho A\frac{\partial^2 v}{\partial t^2}w\mathrm{d}x=\int_0^l\rho A\begin{Bmatrix}N_1\\N_2\\N_3\\N_4\end{Bmatrix}\begin{bmatrix}N_1 & N_2 & N_2 & N_4\end{bmatrix}\mathrm{d}x\begin{Bmatrix}\ddot{v}_1\\\ddot{\theta}_1\\\ddot{v}_2\\\ddot{\theta}_2\end{Bmatrix}\tag{5-11}$$

$$=\int_0^l N^{\mathrm{T}}\rho AN\mathrm{d}x\ddot{\delta}^e=M^e\ddot{\delta}^e$$

$$\int_{\Omega^e}EI\frac{\partial^2 v}{\partial x^2}\frac{\partial^2 w}{\partial x^2}\mathrm{d}x=\int_0^l EI\begin{Bmatrix}N_1''\\N_2''\\N_3''\\N_4''\end{Bmatrix}\begin{bmatrix}N_1'' & N_2'' & N_3'' & N_4''\end{bmatrix}\mathrm{d}x\begin{Bmatrix}v_1\\\theta_1\\v_2\\\theta_2\end{Bmatrix}\tag{5-12}$$

$$=\int_0^l B^{\mathrm{T}}EIB\mathrm{d}x\delta^e=K^e\delta^e$$

$$\int_{\Omega^e}qw\,\mathrm{d}x=\int_0^l q(x,t)\begin{Bmatrix}N_1\\N_2\\N_3\\N_4\end{Bmatrix}\mathrm{d}x=\int_0^l q(x,t)N^{\mathrm{T}}\mathrm{d}x=F^e\tag{5-13}$$

其中

$$M^e=\int_0^l N^{\mathrm{T}}\rho AN\mathrm{d}x\tag{5-14}$$

$$K^e=\int_0^l B^{\mathrm{T}}EI\,B\mathrm{d}x\tag{5-15}$$

$$B=\begin{bmatrix}N_1'' & N_2'' & N_3'' & N_4''\end{bmatrix}\tag{5-16}$$

式(5-14)和式(5-15)分别为单元的质量矩阵和刚度矩阵. 当单元内的质量密度 ρA 和刚度 EI 为常量时，单元质量矩阵和刚度矩阵为

$$M^e=\frac{\rho Al}{420}\begin{bmatrix}156 & 22l & 54 & -13l\\22l & 4l^2 & 13l & -3l^2\\54 & 13l & 156 & -22l\\-13l & -3l^2 & -22l & 4l^2\end{bmatrix}\tag{5-17}$$

$$K^e=\frac{EI}{l^3}\begin{bmatrix}12 & 6l & -12 & 6l\\6l & 4l^2 & -6l & 2l^2\\-12 & -6l & 12 & -6l\\6l & 2l^2 & -6l & 4l^2\end{bmatrix}\tag{5-18}$$

其中，质量矩阵称为一致质量矩阵. 在动力学分析中，为了计算方便，有时也采用集中质

量矩阵

$$M^e = \frac{\rho Al}{2}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{5-19}$$

其为对角阵，便于求逆.

式(5-13)称为单元力向量. 当梁上的作用力 q 为均布载荷 q_0 时，单元力向量为

$$F^e = q_0\int_0^l N^{\mathrm{T}}\mathrm{d}x = \frac{q_0}{12}\begin{bmatrix} 6l & l^2 & 6l & -l^2 \end{bmatrix}^{\mathrm{T}} \tag{5-20}$$

当梁上作用有集中载荷 $P_0(t)$ 时，单元力向量为

$$F^e = \int_0^l P_0(t)\delta(x-x_0)\begin{Bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \end{Bmatrix}\mathrm{d}x = P_0(t)\begin{Bmatrix} N_1(x_0) \\ N_2(x_0) \\ N_3(x_0) \\ N_4(x_0) \end{Bmatrix} \tag{5-21}$$

式中，x_0 为集中载荷 $P_0(t)$ 的作用点坐标；$\delta(x-x_0)$ 是 Dirac delta 函数.

式(5-4)左边的第四项为梁端点剪力和弯矩边界条件，可以包含在系统的力向量中. 因此，将单元矩阵和向量组集后，方程(5-4)变为如下的矩阵方程

$$M\ddot{\delta} + K\delta = F(t) \tag{5-22}$$

称为梁结构的系统运动方程. 求解该方程可给出梁的动态响应.

5.1.2 Timoshenko 梁单元

在 Euler-Bernoulli 梁单元中没有考虑梁的剪切变形. 在工程实际应用中，也常常会遇到需要考虑剪切变形影响的情况. 考虑梁的剪切变形时可以采用 Timoshenko 梁的理论. 对于 Timoshenko 梁，原来垂直于中面的截面变形后不再和中面垂直，截面的转角变为

$$\theta = \frac{\mathrm{d}v}{\mathrm{d}x} - \gamma \tag{5-23}$$

式中，γ 是截面和中面相交处的剪切应变，如图 5.3 所示.

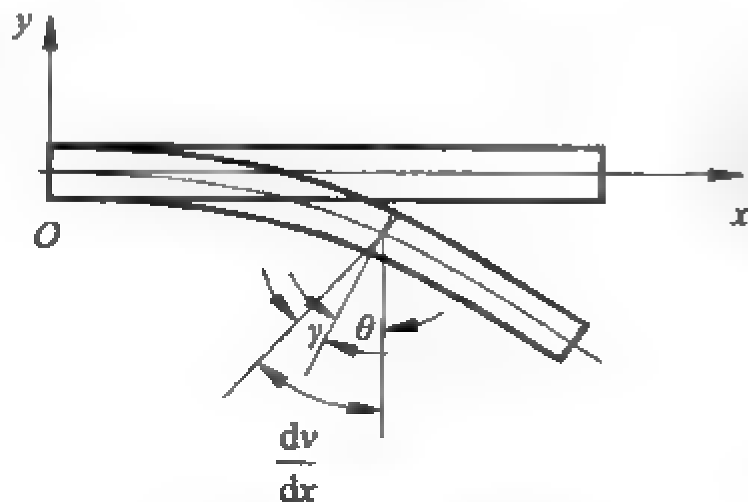


图 5.3 有剪切影响的梁变形

考虑梁任一截面的位移，有

$$u(x,y)=-y\theta(x) \tag{5-24}$$

$$v(x)=v \tag{5-25}$$

式中：$u(x,y)$ 为梁沿轴向的位移.

由式(5-24)和式(5-25)，梁的轴向应变和剪切应变为

$$\varepsilon=-y\frac{\mathrm{d}\theta}{\mathrm{d}x} \tag{5-26}$$

$$\gamma=-\theta+\frac{\mathrm{d}v}{\mathrm{d}x} \tag{5-27}$$

这里用最小势能原理来导出梁的矩阵方程. 梁单元的应变能为

$$U=\frac{1}{2}b\int_0^l\int_{-h/2}^{h/2}\varepsilon^{\mathrm{T}}E\varepsilon\mathrm{d}y\mathrm{d}x+\frac{1}{2}b\int_0^l\int_{-h/2}^{h/2}\gamma^{\mathrm{T}}\frac{G}{k}\gamma\mathrm{d}y\mathrm{d}x \tag{5-28}$$

式(5-28)中右边的第一项为弯曲变形能，第二项为剪切变形能. 式中，b、h和l分别为梁的宽度、高度和长度，k为考虑实际剪切应变和剪切应力不是均匀分布而引入的校正因子，对于矩形截面可取$k=6/5$，对于圆形截面可取$k=10/9$.

将式(5-26)和式(5-27)代入式(5-28)中，可得

$$U=\frac{1}{2}\int_0^l(\frac{\mathrm{d}\theta}{\mathrm{d}x})^{\mathrm{T}}EI(\frac{\mathrm{d}\theta}{\mathrm{d}x})\mathrm{d}x+\frac{1}{2}\int_0^l(-\theta+\frac{\mathrm{d}v}{\mathrm{d}x})^{\mathrm{T}}\frac{GA}{k}(-\theta+\frac{\mathrm{d}v}{\mathrm{d}x})\mathrm{d}x \tag{5-29}$$

其中

$$I=\int_{-h/2}^{h/2}by^2\mathrm{d}y,\quad A=\int_{-h/2}^{h/2}b\mathrm{d}y$$

分别为梁的截面惯性矩和截面面积.

Timoshenko 梁单元中，横向位移v和转角θ是独立的，可各自独立插值，有

$$v=\begin{bmatrix}N_1 & N_2\end{bmatrix}\begin{Bmatrix}v_1\\v_2\end{Bmatrix} \tag{5-30}$$

$$\theta=\begin{bmatrix}N_1 & N_2\end{bmatrix}\begin{Bmatrix}\theta_1\\\theta_2\end{Bmatrix} \tag{5-31}$$

式中，N_1和N_2为线性形函数

$$N_1=\frac{1}{2}(1-\xi),\quad N_2=\frac{1}{2}(1+\xi) \tag{5-32}$$

其中

$$x=N_1(\xi)x_1+N_2(\xi)x_2,\quad \xi=\frac{2x-x_1-x_2}{l} \tag{5-33}$$

将式(5-30)和式(5-31)代入应变能表达式(5-29)，可得

$$\begin{aligned}\frac{1}{2}\int_0^l(\frac{\mathrm{d}\theta}{\mathrm{d}x})^{\mathrm{T}}EI(\frac{\mathrm{d}\theta}{\mathrm{d}x})\mathrm{d}x&=\frac{1}{2}[\theta_1\quad\theta_2]\int_0^l\begin{Bmatrix}N_1'\\N_2'\end{Bmatrix}EI[N_1'\quad N_2']\mathrm{d}x\begin{Bmatrix}\theta_1\\\theta_2\end{Bmatrix}\\&=\frac{1}{2}\boldsymbol{\delta}^{e\mathrm{T}}(\int_0^l\boldsymbol{B}^{\mathrm{T}}EI\,\boldsymbol{B}\mathrm{d}x)\boldsymbol{\delta}^e=\frac{1}{2}\boldsymbol{\delta}^{e\mathrm{T}}\boldsymbol{K}_b^e\boldsymbol{\delta}^e\end{aligned} \tag{5-34}$$

$$
\begin{aligned}
\frac{1}{2}\int_0^l \left(-\theta+\frac{\mathrm{d}v}{\mathrm{d}x}\right)^{\mathrm{T}}\frac{GA}{k}\left(-\theta+\frac{\mathrm{d}v}{\mathrm{d}x}\right)\mathrm{d}x &= \frac{1}{2}[\theta_1 \quad \theta_2]\int_0^l \begin{Bmatrix} N_1 \\ N_2 \end{Bmatrix}\frac{GA}{k}[N_1 \quad N_2]\mathrm{d}x\begin{Bmatrix} \theta_1 \\ \theta_2 \end{Bmatrix} \\
&\quad -\frac{1}{2}[\theta_1 \quad \theta_2]\int_0^l \begin{Bmatrix} N_1 \\ N_2 \end{Bmatrix}\frac{GA}{k}[N_1' \quad N_2']\mathrm{d}x\begin{Bmatrix} v_1 \\ v_2 \end{Bmatrix} \\
&\quad -\frac{1}{2}[v_1 \quad v_2]\int_0^l \begin{Bmatrix} N_1' \\ N_2' \end{Bmatrix}\frac{GA}{k}[N_1 \quad N_2]\mathrm{d}x\begin{Bmatrix} \theta_1 \\ \theta_2 \end{Bmatrix} \\
&\quad +\frac{1}{2}[v_1 \quad v_2]\int_0^l \begin{Bmatrix} N_1' \\ N_2' \end{Bmatrix}\frac{GA}{k}[N_1' \quad N_2']\mathrm{d}x\begin{Bmatrix} v_1 \\ v_2 \end{Bmatrix} \qquad (5\text{-}35) \\
&= \frac{1}{2}\boldsymbol{\delta}^{e\mathrm{T}}\Big[\int_0^l \boldsymbol{N}^{\mathrm{T}}\frac{GA}{k}\boldsymbol{N}\mathrm{d}x - \int_0^l \boldsymbol{N}^{\mathrm{T}}\frac{GA}{k}\bar{\boldsymbol{B}}\mathrm{d}x \\
&\quad - \int_0^l \bar{\boldsymbol{B}}^{\mathrm{T}}\frac{GA}{k}\boldsymbol{N}\mathrm{d}x + \int_0^l \bar{\boldsymbol{B}}^{\mathrm{T}}\frac{GA}{k}\bar{\boldsymbol{B}}\mathrm{d}x\Big]\boldsymbol{\delta}^e \\
&= \frac{1}{2}\boldsymbol{\delta}^{e\mathrm{T}}\boldsymbol{K}_s^e\boldsymbol{\delta}^e
\end{aligned}
$$

其中

$$
\boldsymbol{\delta}^e = \begin{Bmatrix} v_1 \\ \theta_1 \\ v_2 \\ \theta_2 \end{Bmatrix}, \quad \boldsymbol{N} = [0 \quad N_1 \quad 0 \quad N_2],
$$

$$
\boldsymbol{B} = [0 \quad N_1' \quad 0 \quad N_2'], \quad \bar{\boldsymbol{B}} = [N_1' \quad 0 \quad N_2' \quad 0]
$$

由此可知，Timoshenko 梁单元的刚度矩阵为

$$
\boldsymbol{K}^e = \boldsymbol{K}_b^e + \boldsymbol{K}_s^e \qquad (5\text{-}36)
$$

再将形函数式(5-32)代入式(5-34)和式(5-35)，可得弯曲刚度矩阵和剪切刚度矩阵

$$
\boldsymbol{K}_b^e = \frac{EI}{l}\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \qquad (5\text{-}37)
$$

$$
\boldsymbol{K}_s^e = \frac{GA}{6kl}\begin{bmatrix} 6 & 3l & -6 & 3l \\ 3l & 2l^2 & -3l & l^2 \\ -6 & -3l & 6 & -3l \\ 3l & l^2 & -3l & 2l^2 \end{bmatrix} \qquad (5\text{-}38)
$$

上述的 Timoshenko 梁刚度矩阵用于薄梁时会产生误差. 由式(5-37)和式(5-38)可知，弯曲刚度矩阵与 h^3/l 成正比，而剪切刚度矩阵则与 h/l 成正比. 当 h/l 很小时，将导致弯曲变形项可忽略不计，这与实际情况不符. 造成这种现象的原因是横向位移 v 和转动 θ 采用同阶插值表达式，使得剪切应变 γ 中的 $\frac{\mathrm{d}v}{\mathrm{d}x}$ 和 θ 是不同阶的，这样一来约束条件 $\gamma-\frac{\mathrm{d}v}{\mathrm{d}x}-\theta=0$ 不可能处处满足，这种现象称为剪切锁死(shear locking). 为了避免这种剪切

锁死可以采用减缩积分(reduced integration)的方法，即在数值积分时用比精确积分要求少的积分点数．对于二节点单元，在积分剪切应变能时用一点高斯积分，有

$$\boldsymbol{K}_s^e=\frac{GA}{4kl}\begin{bmatrix}4 & 2l & -4 & 2l\\ 2l & l^2 & -2l & l^2\\ -4 & -2l & 4 & -2l\\ 2l & l^2 & -2l & l^2\end{bmatrix} \tag{5-39}$$

Timoshenko 梁的质量矩阵可由动能表达式得到．单元的动能可表示为

$$T=\frac{1}{2}\int_0^l\int_{-h/2}^{h/2}\dot{v}^{\mathrm{T}}\rho\dot{v}b\mathrm{d}y\mathrm{d}x=\frac{1}{2}\int_0^l\dot{v}^{\mathrm{T}}\rho A\dot{v}\mathrm{d}x \tag{5-40}$$

式中，ρ 为单元的质量密度．

将式(5-30)代入式(5-40)中，得

$$\begin{aligned}T&=\frac{1}{2}\int_0^l\begin{bmatrix}\dot{v}_1 & \dot{v}_2\end{bmatrix}\begin{Bmatrix}N_1\\ N_2\end{Bmatrix}\rho A\begin{bmatrix}N_1 & N_2\end{bmatrix}\begin{Bmatrix}\dot{v}_1\\ \dot{v}_2\end{Bmatrix}\mathrm{d}x\\ &=\frac{1}{2}\begin{bmatrix}\dot{v}_1 & \dot{\theta}_1 & \dot{v}_2 & \dot{\theta}_2\end{bmatrix}\left(\int_0^l\begin{Bmatrix}N_1\\ 0\\ N_2\\ 0\end{Bmatrix}\rho A\begin{bmatrix}N_1 & 0 & N_2 & 0\end{bmatrix}\mathrm{d}x\right)\begin{Bmatrix}\dot{v}_1\\ \dot{\theta}_1\\ \dot{v}_2\\ \dot{\theta}_2\end{Bmatrix}\\ &=\frac{1}{2}\dot{\boldsymbol{\delta}}^{e\mathrm{T}}\left(\int_0^l\boldsymbol{N}^{\mathrm{T}}\rho A\boldsymbol{N}\mathrm{d}x\right)\dot{\boldsymbol{\delta}}^e=\frac{1}{2}\dot{\boldsymbol{\delta}}^{e\mathrm{T}}\boldsymbol{M}^e\dot{\boldsymbol{\delta}}^e\end{aligned} \tag{5-41}$$

其中

$$\boldsymbol{N}=\begin{bmatrix}N_1 & 0 & N_2 & 0\end{bmatrix}=\begin{bmatrix}\frac{1}{2}(1-\xi) & 0 & \frac{1}{2}(1+\xi) & 0\end{bmatrix}$$

$$\boldsymbol{\delta}^e=\begin{bmatrix}v_1 & \theta_1 & v_2 & \theta_2\end{bmatrix}^{\mathrm{T}}$$

于是 Timoshenko 梁的单元质量矩阵为

$$\boldsymbol{M}^e=\int_0^l\boldsymbol{N}^{\mathrm{T}}\rho A\boldsymbol{N}\mathrm{d}x=\frac{\rho Al}{6}\begin{bmatrix}2 & 0 & 1 & 0\\ 0 & 0 & 0 & 0\\ 1 & 0 & 2 & 0\\ 0 & 0 & 0 & 0\end{bmatrix} \tag{5-42}$$

Timoshenko 梁单元的集中质量矩阵与式(5-19)相同．Timoshenko 梁单元的力向量计算与 Euler-Bernoulli 梁相同．

将单元矩阵和向量组集后，梁结构的矩阵方程为

$$\boldsymbol{M}\ddot{\boldsymbol{\delta}}+\boldsymbol{K}\boldsymbol{\delta}=\boldsymbol{F}(t) \tag{5-43}$$

称为梁结构的系统运动方程．求解该方程可给出梁的动态响应．

5.1.3 考虑剪切变形的 Euler-Bernoulli 梁单元

在 Euler-Bernoulli 梁的假设中忽略了剪切变形的作用．计入剪切变形的一种方法是对 Euler-Bernoulli 梁进行修正，引入剪切变形的影响．

考虑剪切变形影响时，梁的横向位移可表示为

$$v = v_b + v_s \tag{5-44}$$

式中：v_b 是由弯曲变形引起的横向位移；v_s 是由剪切变形引起的横向位移. 相应地，单元的节点变量也表示成两部分 $\boldsymbol{\delta}_b^e$ 和 $\boldsymbol{\delta}_s^e$

$$\boldsymbol{\delta}_b^e = \begin{bmatrix} v_{b1} & \theta_1 & v_{b2} & \theta_2 \end{bmatrix}^{\mathrm{T}}, \boldsymbol{\delta}_s^e = \begin{bmatrix} v_{s1} & v_{s2} \end{bmatrix}^{\mathrm{T}} \tag{5-45}$$

式中，$\theta_1 = \left(\dfrac{\mathrm{d}v_b}{\mathrm{d}x}\right)_1$，$\theta_2 = \left(\dfrac{\mathrm{d}v_b}{\mathrm{d}x}\right)_2$，即 v_b 仍采用和不考虑剪切变形时的位移，可用 Hermite 形函数表示. 而 v_s 中只有两个节点参数，可用 Lagrange 形函数表示，即有

$$\begin{aligned} v_b(x) &= N_1 v_{b1} + N_2\theta_1 + N_3 v_{b2} + N_4\theta_4 = \boldsymbol{N}_b \boldsymbol{\delta}_b^e \\ v_s(x) &= N_5 v_{s1} + N_6 v_{s2} = \boldsymbol{N}_s \boldsymbol{\delta}_s^e \end{aligned} \tag{5-46}$$

其中

$$\boldsymbol{N}_b = \begin{bmatrix} N_1 & N_2 & N_3 & N_4 \end{bmatrix}, \quad \boldsymbol{N}_s = \begin{bmatrix} N_5 & N_6 \end{bmatrix}$$

$$N_1 = 1 - 3\xi^2 + 2\xi^3, \quad N_2(x) = (\xi - 2\xi^2 + \xi^3)l, \quad N_3(x) = 3\xi^2 - 2\xi^3,$$

$$N_4(x) = (-\xi^2 + \xi^3)l, N_5 = 1 - \xi, \quad N_6 = \xi$$

$$\xi = \frac{x - x_1}{l} \tag{5-47}$$

将式(5-45)和式(5-46)代入式(5-29)表示的应变能表达式

$$U = \frac{1}{2}\int_0^l \left(\frac{\mathrm{d}\theta}{\mathrm{d}x}\right)^{\mathrm{T}} EI\left(\frac{\mathrm{d}\theta}{\mathrm{d}x}\right)\mathrm{d}x + \frac{1}{2}\int_0^l \gamma^{\mathrm{T}} \frac{GA}{k}\gamma \mathrm{d}x$$

得

$$\begin{aligned} \frac{1}{2}\int_0^l \left(\frac{\mathrm{d}\theta}{\mathrm{d}x}\right)^{\mathrm{T}} EI\left(\frac{\mathrm{d}\theta}{\mathrm{d}x}\right)\mathrm{d}x &= \frac{1}{2}\int_0^l \left(\frac{\mathrm{d}^2 v_b}{\mathrm{d}x^2}\right)^{\mathrm{T}} EI\left(\frac{\mathrm{d}^2 v_b}{\mathrm{d}x^2}\right)\mathrm{d}x = \frac{1}{2}\boldsymbol{\delta}_b^{e\mathrm{T}}\left(\int_0^l \boldsymbol{N}_b''^{\mathrm{T}} EI\, \boldsymbol{N}_b''\mathrm{d}x\right)\boldsymbol{\delta}_b^e \\ &= \frac{1}{2}\boldsymbol{\delta}_b^{e\mathrm{T}}\left(\int_0^l \boldsymbol{B}_b^{\mathrm{T}} EI\, \boldsymbol{B}_b \mathrm{d}x\right)\boldsymbol{\delta}_b^e = \frac{1}{2}\boldsymbol{\delta}_b^{e\mathrm{T}}\boldsymbol{K}_b^e\boldsymbol{\delta}_b^e \end{aligned} \tag{5-48}$$

$$\begin{aligned} \frac{1}{2}\int_0^l \gamma^{\mathrm{T}} \frac{GA}{k}\gamma \mathrm{d}x &= \frac{1}{2}\int_0^l \left(\frac{\mathrm{d}v_s}{\mathrm{d}x}\right)^{\mathrm{T}} \frac{GA}{k}\left(\frac{\mathrm{d}v_s}{\mathrm{d}x}\right)\mathrm{d}x = \frac{1}{2}\boldsymbol{\delta}_s^{e\mathrm{T}}\left(\int_0^l \boldsymbol{N}_s'^{\mathrm{T}} \frac{GA}{k}\boldsymbol{N}_s'\mathrm{d}x\right)\boldsymbol{\delta}_s^e \\ &= \frac{1}{2}\boldsymbol{\delta}_s^{e\mathrm{T}}\left(\int_0^l \boldsymbol{B}_s^{\mathrm{T}} \frac{GA}{k}\boldsymbol{B}_s \mathrm{d}x\right)\boldsymbol{\delta}_s^e = \frac{1}{2}\boldsymbol{\delta}_s^{e\mathrm{T}}\boldsymbol{K}_s^e\boldsymbol{\delta}_s^e \end{aligned} \tag{5-49}$$

其中

$$\boldsymbol{B}_b = \begin{bmatrix} N_1'' & N_2'' & N_3'' & N_4'' \end{bmatrix}, \quad \boldsymbol{B}_s = \begin{bmatrix} N_5' & N_6' \end{bmatrix}$$

再将式(5-47)表示的形函数代入式(5-48)和(5-49)，可得单元的弯曲刚度矩阵和剪切刚度矩阵

$$\boldsymbol{K}_b^e = \frac{EI}{l^3}\begin{bmatrix} 12 & 6l & -12 & 6l \\ 6l & 4l^2 & -6l & 2l^2 \\ -12 & -6l & 12 & -6l \\ 6l & 2l^2 & -6l & 4l^2 \end{bmatrix} \tag{5-50}$$

$$\boldsymbol{K}_s^e = \frac{GA}{kl}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{5-51}$$

下面计算梁单元的质量矩阵. 考虑剪切变形影响时梁单元的动能

$$T=\frac{1}{2}\int_0^l\int_{-h/2}^{h/2}\dot{v}^{\mathrm{T}}\rho\dot{v}b\mathrm{d}y\mathrm{d}x=\frac{1}{2}\int_0^l\dot{v}^{\mathrm{T}}\rho A\dot{v}\mathrm{d}x=\frac{1}{2}\int_0^l\dot{v}_b\rho A\dot{v}_b\mathrm{d}x+\frac{1}{2}\int_0^l\dot{v}_s\rho A\dot{v}_s\mathrm{d}x \tag{5-52}$$

式中：b、h和l分别是梁的宽度、高度和长度；ρ为质量密度.

将式(5-45)和式(5-46)代入式(5-52)，得

$$\frac{1}{2}\int_0^l\dot{v}_b\rho A\dot{v}_b\mathrm{d}x=\frac{1}{2}\begin{bmatrix}\dot{v}_{b1} & \dot{\theta}_1 & \dot{v}_{b2} & \dot{\theta}_2\end{bmatrix}\left(\int_0^l\begin{Bmatrix}N_1\\N_2\\N_3\\N_4\end{Bmatrix}\rho A\begin{bmatrix}N_1 & N_2 & N_3 & N_4\end{bmatrix}\mathrm{d}x\right)\begin{Bmatrix}\dot{v}_{b1}\\\dot{\theta}_1\\\dot{v}_{b2}\\\dot{\theta}_2\end{Bmatrix} \tag{5-53}$$

$$=\frac{1}{2}\dot{\delta}_b^{e\mathrm{T}}\left(\int_0^l N_b^{\mathrm{T}}\rho A N_b\mathrm{d}x\right)\dot{\delta}_b^e=\frac{1}{2}\dot{\delta}_b^{e\mathrm{T}}M_b^e\dot{\delta}_b^e$$

$$\frac{1}{2}\int_0^l\dot{v}_s\rho A\dot{v}_s\mathrm{d}x=\frac{1}{2}\begin{bmatrix}\dot{v}_{s1} & \dot{v}_{s2}\end{bmatrix}\left(\int_0^l\begin{Bmatrix}N_5\\N_6\end{Bmatrix}\rho A\begin{bmatrix}N_5 & N_6\end{bmatrix}\mathrm{d}x\right)\begin{Bmatrix}\dot{v}_{s1}\\\dot{v}_{s2}\end{Bmatrix}$$
$$=\frac{1}{2}\dot{\delta}_s^{e\mathrm{T}}\left(\int_0^l N_s^{\mathrm{T}}\rho A N_s\mathrm{d}x\right)\dot{\delta}_s^e=\frac{1}{2}\dot{\delta}_s^{e\mathrm{T}}M_s^e\dot{\delta}_s^e \tag{5-54}$$

再将式(5-47)表示的形函数代入式(5-53)和(5-54)，可得单元的质量矩阵

$$M_b^e=\frac{\rho Al}{420}\begin{bmatrix}156 & 22l & 54 & -13l\\22l & 4l^2 & 13l & -3l^2\\54 & 13l & 156 & -22l\\-13l & -3l^2 & -22l & 4l^2\end{bmatrix} \tag{5-55}$$

$$M_s^e=\frac{\rho Al}{6}\begin{bmatrix}2 & 1\\1 & 2\end{bmatrix} \tag{5-56}$$

梁单元的力向量计算与Euler-Bernoulli梁相同.

将单元矩阵和向量组集后，梁结构的矩阵方程为

$$\begin{cases}M_b\ddot{\delta}+K_b\delta=F_b(t)\\M_s\ddot{\delta}+K_s\delta=F_s(t)\end{cases} \tag{5-57}$$

称为梁结构的系统运动方程.求解该方程可给出梁的动态响应.

在上述方程中，每个节点有3个位移变量：v_{bi}，v_{si}，θ_i. 实际计算中，利用单元的平衡方程和几何关系可以消去一个变量，可得到两个独立变量v_i和θ_i的方程[1]. 此时，单元刚度矩阵为

$$K^e=\frac{EI}{(1+b)l^3}\begin{bmatrix}12 & 6l & -12 & 6l\\6l & (4+b)l^2 & -6l & (2-b)l^2\\-12 & -6l & 12 & -6l\\6l & (2-b)l^2 & -6l & (4+b)l^2\end{bmatrix} \tag{5-58}$$

式中，$b=\dfrac{12kEI}{GAl^2}$.

5.1.4　混合梁单元

混合梁单元是同时将位移和力作为变量的有限元法，有利于提高梁应力计算的精度[5]. 由梁变形时的挠度曲率与弯矩的关系及载荷与内力的关系，有

$$\boldsymbol{M} = EI\frac{\mathrm{d}^2 v}{\mathrm{d}x^2} \tag{5-59}$$

$$\frac{\mathrm{d}^2 \boldsymbol{M}}{\mathrm{d}x^2} = q \tag{5-60}$$

式中：E 是材料杨氏弹性模量；I 是梁横截面的面积矩.

用加权余量法(weighted residual method)，上述方程的残差为

$$\int_0^L \left(\boldsymbol{M} + EI\frac{\mathrm{d}^2 v}{\mathrm{d}x^2}\right) w_1 \mathrm{d}x = 0 \tag{5-61}$$

$$\int_0^L \left(\frac{\mathrm{d}^2 \boldsymbol{M}}{\mathrm{d}x^2} - q\right) w_2 \mathrm{d}x = 0 \tag{5-62}$$

式中：L 为梁的长度；w_i 为试探函数.

经过分部积分，式(5-61)和式(5-62)变为

$$\frac{1}{EI}\int_0^L \boldsymbol{M} w_1 \mathrm{d}x + \int_0^L \frac{\mathrm{d}v}{\mathrm{d}x}\frac{\mathrm{d}w_1}{\mathrm{d}x}\mathrm{d}x - [w_1\theta]_0^L = 0 \tag{5-63}$$

$$\int_0^L \frac{\mathrm{d}\boldsymbol{M}}{\mathrm{d}x} w_2 \mathrm{d}x + \int_0^L q w_2 \mathrm{d}x - [w_2 V]_0^L = 0 \tag{5-64}$$

式中：$\theta = \dfrac{\mathrm{d}v}{\mathrm{d}x}$ 为梁的转角；$V = \dfrac{\mathrm{d}M}{\mathrm{d}x}$ 为梁的剪力.

将梁离散化后，上述方程变为

$$\sum_{i=1}^{n}\left[\frac{1}{EI}\int_{\Omega^e} M w_1 \mathrm{d}x + \int_{\Omega^e} \frac{\mathrm{d}v}{\mathrm{d}x}\frac{\mathrm{d}w_1}{\mathrm{d}x}\mathrm{d}x\right] - [w_1\theta]_0^L = 0 \tag{5-65}$$

$$\sum_{i=1}^{n}\left[\int_{\Omega^e} \frac{\mathrm{d}M}{\mathrm{d}x} w_2 \mathrm{d}x + \int_{\Omega^e} q w_2 \mathrm{d}x\right] - [w_2 V]_0^L = 0 \tag{5-66}$$

式中：$\boldsymbol{\Omega}$ 为单元的定义域；n 为单元的数目.

单元的节点变量为

$$\boldsymbol{\delta}^e = [M_1 \quad M_2 \quad v_1 \quad v_2]^{\mathrm{T}} \tag{5-67}$$

对 $\boldsymbol{M}$ 和 v 取相同的形函数进行插值

$$\boldsymbol{M} = N_1 M_1 + N_2 M_2 \tag{5-68}$$

$$v = N_1 v_1 + N_2 v_2 \tag{5-69}$$

运用 Galerkin 方法，将 w_1、w_2 取为与 $\boldsymbol{M}$ 和 v 相同的形函数，对于一个单元有

$$\frac{1}{EI}\int_0^l \boldsymbol{N}^{\mathrm{T}}\boldsymbol{N}\mathrm{d}x\begin{bmatrix} M_1 \\ M_2 \end{bmatrix} + \int_0^l \left(\frac{\mathrm{d}\boldsymbol{N}}{\mathrm{d}x}\right)^{\mathrm{T}}\frac{\mathrm{d}\boldsymbol{N}}{\mathrm{d}x}\mathrm{d}x\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = [\boldsymbol{N}^{\mathrm{T}}\theta]_0^l \tag{5-70}$$

$$\int_0^l \left(\frac{\mathrm{d}\boldsymbol{N}}{\mathrm{d}x}\right)^{\mathrm{T}}\frac{\mathrm{d}\boldsymbol{N}}{\mathrm{d}x}\mathrm{d}x\begin{bmatrix} M_1 \\ M_2 \end{bmatrix} = -\int_0^l \boldsymbol{N}^{\mathrm{T}} q \mathrm{d}x + [\boldsymbol{N}^{\mathrm{T}} V]_0^l \tag{5-71}$$

其中

$$N=\begin{bmatrix}N_1 & N_2\end{bmatrix} \tag{5-72}$$

于是，单元的平衡方程为

$$K^e\delta^e=F^e \tag{5-73}$$

其中

$$K^e=\begin{bmatrix}K_{11}^e & K_{12}^e\\ K_{21}^e & 0\end{bmatrix} \tag{5-74}$$

$$K_{11}^e=\frac{1}{EI}\int_0^l N^{\mathrm T}N\mathrm dx\text{，}\quad K_{12}^e=\int_0^l\left(\frac{\mathrm dN}{\mathrm dx}\right)^{\mathrm T}\frac{\mathrm dN}{\mathrm dx}\mathrm dx\text{，}\quad K_{21}^e=K_{12}^{e\,\mathrm T} \tag{5-75}$$

$$F^e=\begin{bmatrix}\theta_1 & \theta_2 & V_1-Q_1 & V_2-Q_2\end{bmatrix}^{\mathrm T} \tag{5-76}$$

$$\begin{bmatrix}Q_1\\ Q_2\end{bmatrix}=\int_0^l N^{\mathrm T}q\mathrm dx \tag{5-77}$$

对于线性混合梁单元(如图 5.4 所示)，N_1 和 N_2 可取为线性形函数

$$N_1=\frac{1}{2}(1-\xi)\text{，}\quad N_2=\frac{1}{2}(1+\xi) \tag{5-78}$$

其中

$$\xi=\frac{2x-x_1-x_2}{l}$$

则有

$$K^e=\frac{1}{6EIl}\begin{bmatrix}2l^2 & l^2 & 6EI & -6EI\\ l^2 & 2l^2 & -6EI & 6EI\\ 6EI & -6EI & 0 & 0\\ -6EI & 6EI & 0 & 0\end{bmatrix} \tag{5-79}$$

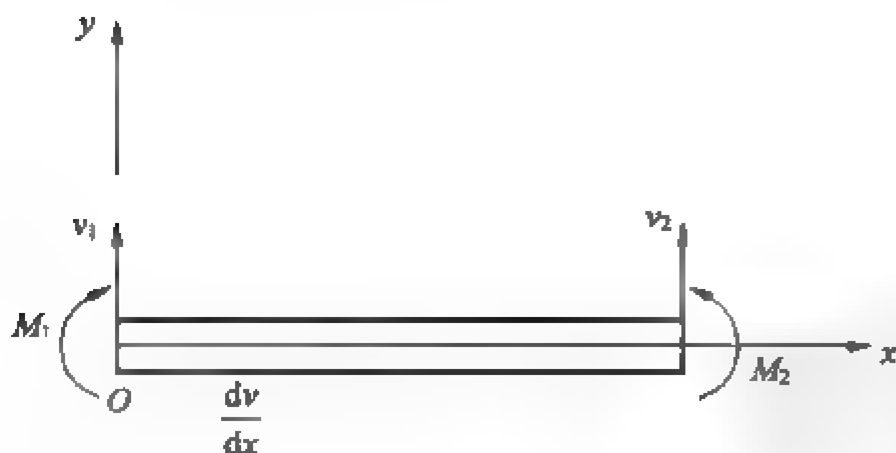


图 5.4　线性混合梁单元

当考虑梁剪切变形的影响时，梁变形的控制方程变为

$$\frac{M}{EI}-\frac{k}{GA}\frac{\mathrm d^2M}{\mathrm dx^2}-\frac{\mathrm d^2v}{\mathrm dx^2}=0 \tag{5-80}$$

$$\frac{\mathrm d^2M}{\mathrm dx^2}-q \tag{5-81}$$

式中：G 是材料剪切弹性模量；A 是梁横截面面积；k 为考虑实际剪切应变和剪切应力不

是均匀分布而引入的校正因子.

同样采用 Galerkin 方法，得到单元刚度矩阵的计算式为

$$\boldsymbol{K}^e = \begin{bmatrix} K_{11}^e & K_{12}^e \\ K_{21}^e & 0 \end{bmatrix} \tag{5-82}$$

其中

$$K_{11}^e = \frac{1}{EI}\int_0^l N^T N \mathrm{d}x + \frac{k}{GA}\int_0^l \left(\frac{\mathrm{d}N}{\mathrm{d}x}\right)^T \frac{\mathrm{d}N}{\mathrm{d}x}\mathrm{d}x$$

$$K_{12}^e = \int_0^l \left(\frac{\mathrm{d}N}{\mathrm{d}x}\right)^T \frac{\mathrm{d}N}{\mathrm{d}x}\mathrm{d}x, \quad K_{21}^e = K_{12}^{e\,T} \tag{5-83}$$

当 M 和 v 的形函数为线性函数时，单元刚度矩阵为

$$\boldsymbol{K}^e = \frac{1}{6EIl}\begin{bmatrix} 2l^2+a & l^2-a & 6EI & -6EI \\ l^2-a & 2l^2+a & -6EI & 6EI \\ 6EI & -6EI & 0 & 0 \\ -6EI & 6EI & 0 & 0 \end{bmatrix} \tag{5-84}$$

其中

$$a = \frac{6EIk}{GA} \tag{5-85}$$

对于线性混合梁单元的集中质量矩阵为

$$\boldsymbol{M}^e = \frac{\rho Al}{2}\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5-86}$$

5.2 平面刚架

刚架结构系统是由多个梁构件组成的结构系统. 如果组成刚架结构的梁构件在几何上处于同一平面，并且结构所受的载荷也在该平面内，则称为平面刚架系统. 刚架结构可以用杆-梁单元进行离散化. 一般来说，单元不再是单独地承受拉压、或扭转、或弯曲载荷，而是以它们的共同作用来工作，单元的特性矩阵将是几种单元特性矩阵的组合. 对于平面刚架，单元通常承受轴向拉压和弯曲载荷. 此外，刚架的构件或单元一般不处于同一轴线上，进行分析计算时，需要建立一个共用的总体坐标系，通过坐标转换将单元局部坐标系下的特性矩阵变换到总体坐标系.

平面刚架的单元为二维单元，一般承受轴向力和弯矩的作用，单元的特性是轴力单元和弯曲单元特性的组合. 在小变形的情况下，这两种变形间的耦合可以忽略，因而，平面刚架单元的特性矩阵可以由轴力单元和弯曲单元的特性矩阵叠加来构成.

对于二节点单元，在单元局部坐标系下，节点变量可表示为

$$\boldsymbol{\delta}^e = \{u_1 \quad v_1 \quad \theta_1 \quad u_2 \quad v_2 \quad \theta_2\}^T \tag{5-87}$$

式中：u_i、v_i为节点i沿局部坐标方向的位移；θ_i为节点i处截面的转角.

弯曲单元采用Euler-Bernoulli梁的情况下，二维刚架单元的刚度矩阵为

$$\boldsymbol{K}^e=\begin{bmatrix} \frac{EA}{l} & 0 & 0 & -\frac{EA}{l} & 0 & 0 \\ 0 & \frac{12EI}{l^3} & \frac{6EI}{l^2} & 0 & -\frac{12EI}{l^3} & \frac{6EI}{l^2} \\ 0 & \frac{6EI}{l^2} & \frac{4EI}{l} & 0 & -\frac{6EI}{l^2} & \frac{2EI}{l} \\ -\frac{EA}{l} & 0 & 0 & \frac{EA}{l} & 0 & 0 \\ 0 & -\frac{12EI}{l^3} & -\frac{6EI}{l^2} & 0 & \frac{12EI}{l^3} & -\frac{6EI}{l^2} \\ 0 & \frac{6EI}{l^2} & \frac{2EI}{l} & 0 & -\frac{6EI}{l^2} & \frac{4EI}{l} \end{bmatrix} \tag{5-88}$$

质量矩阵为

$$\boldsymbol{M}^e=\frac{\rho Al}{420}\begin{bmatrix} 140 & 0 & 0 & 70 & 0 & 0 \\ 0 & 156 & 22l & 0 & 54 & -13l \\ 0 & 22l & 4l^2 & 0 & 13l & -3l^2 \\ 70 & 0 & 0 & 140 & 0 & 0 \\ 0 & 54 & 13l & 0 & 156 & -22l \\ 0 & -13l & -3l^2 & 0 & -22l & 4l^2 \end{bmatrix} \tag{5-89}$$

集中质量矩阵为

$$\boldsymbol{M}^e=\frac{\rho Al}{2}\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5-90}$$

采用其他类型的梁单元，可构成不同的二维刚架单元特性矩阵. 若用Timoshenko梁，则刚度矩阵为

$$\boldsymbol{K}^e=\begin{bmatrix} \frac{EA}{l} & 0 & 0 & -\frac{EA}{l} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{EI}{l} & 0 & 0 & -\frac{EI}{l} \\ -\frac{EA}{l} & 0 & 0 & \frac{EA}{l} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{EI}{l} & 0 & 0 & \frac{EI}{l} \end{bmatrix}+\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{GA}{kl} & \frac{GA}{2k} & 0 & -\frac{GA}{kl} & \frac{GA}{2k} \\ 0 & \frac{GA}{2k} & \frac{GAl}{4k} & 0 & -\frac{GA}{2k} & \frac{GAl}{4k} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{GA}{kl} & -\frac{GA}{2k} & 0 & \frac{GA}{kl} & -\frac{GA}{2k} \\ 0 & \frac{GA}{2k} & \frac{GAl}{4k} & 0 & -\frac{GA}{2k} & \frac{GAl}{4k} \end{bmatrix}$$

$$
=\begin{bmatrix}
\frac{EA}{l} & 0 & 0 & -\frac{EA}{l} & 0 & 0 \\
0 & \frac{GA}{kl} & \frac{GA}{2k} & 0 & -\frac{GA}{kl} & \frac{GA}{2k} \\
0 & \frac{GA}{2k} & \frac{GAl}{4k}+\frac{EI}{l} & 0 & -\frac{GA}{2k} & \frac{GAl}{4k}-\frac{EI}{l} \\
-\frac{EA}{l} & 0 & 0 & \frac{EA}{l} & 0 & 0 \\
0 & -\frac{GA}{kl} & -\frac{GA}{2k} & 0 & \frac{GA}{kl} & -\frac{GA}{2k} \\
0 & \frac{GA}{2k} & \frac{GAl}{4k}-\frac{EI}{l} & 0 & -\frac{GA}{2k} & \frac{GAl}{4k}+\frac{EI}{l}
\end{bmatrix} \tag{5-91}
$$

质量矩阵为

$$
M^e=\frac{\rho Al}{6}\begin{bmatrix}
2 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}+\frac{\rho Al}{6}\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 2 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

$$
=\frac{\rho Al}{6}\begin{bmatrix}
2 & 0 & 0 & 1 & 0 & 0 \\
0 & 2 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 2 & 0 & 0 \\
0 & 1 & 0 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \tag{5-92}
$$

而集中质量矩阵与式(5-90)相同.

如前所述，由于刚架系统中单元的局部坐标方向各异，单元的特性矩阵需要通过平面坐标变换转换到总体坐标系上. 如图 5.5 所示为总体坐标系中的平面刚架单元. 其中，总体坐标系用 $\bar{x},\bar{y}$ 表示；局部坐标系用 x,y 表示.

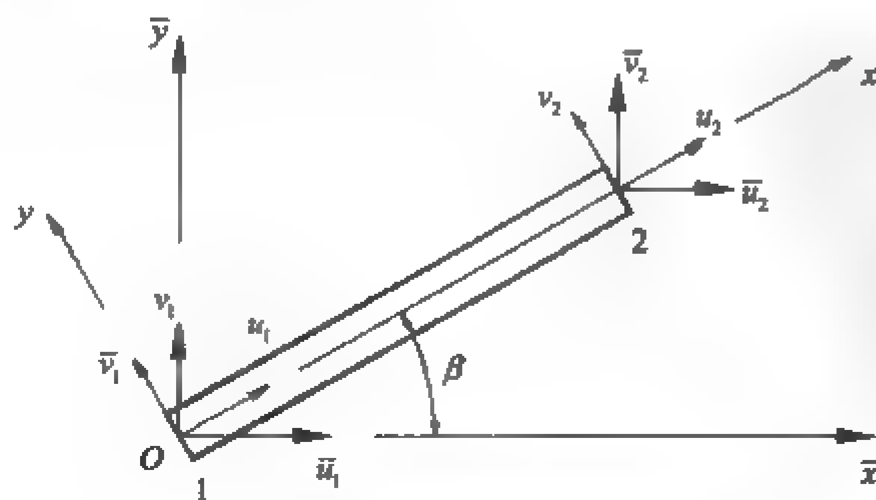


图 5.5　总体坐标系与局部坐标系

由图 5.5 可见，局部坐标系与总体坐标系的关系为

$$\begin{Bmatrix} x \\ y \\ \theta \end{Bmatrix} = \begin{bmatrix} c & s & 0 \\ -s & c & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} \bar{x} \\ \bar{y} \\ \bar{\theta} \end{Bmatrix} \tag{5-93}$$

式中，$c=\cos\beta$；$s=\sin\beta$. 其中，β 为局部坐标系相应于总体坐标系 $\bar{x}$ 轴的夹角.

因而，局部坐标系下的单元节点位移与总体坐标系中的单元节点位移的关系为

$$\begin{Bmatrix} u_1 \\ v_1 \\ \theta_1 \\ u_2 \\ v_2 \\ \theta_2 \end{Bmatrix} = \begin{bmatrix} c & s & 0 & 0 & 0 & 0 \\ -s & c & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & c & s & 0 \\ 0 & 0 & 0 & -s & c & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} \bar{u}_1 \\ \bar{v}_1 \\ \bar{\theta}_1 \\ \bar{u}_2 \\ \bar{v}_2 \\ \bar{\theta}_2 \end{Bmatrix} \tag{5-94}$$

写成矩阵形式，有

$$\boldsymbol{\delta}^e = \boldsymbol{T}\bar{\boldsymbol{\delta}}^e \tag{5-95}$$

其中

$$\boldsymbol{T} = \begin{bmatrix} c & s & 0 & 0 & 0 & 0 \\ -s & c & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & c & s & 0 \\ 0 & 0 & 0 & -s & c & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5-96}$$

将式(5-96)代入有限元的方程中，并对方程前乘 $\boldsymbol{T}^{\mathrm{T}}$，即可得到转换到总体坐标系下的单元特性矩阵

$$\bar{\boldsymbol{K}}^e = \boldsymbol{T}^{\mathrm{T}}\boldsymbol{K}^e\boldsymbol{T}, \quad \bar{\boldsymbol{M}}^e = \boldsymbol{T}^{\mathrm{T}}\boldsymbol{M}^e\boldsymbol{T} \tag{5-97}$$

和力向量

$$\bar{\boldsymbol{F}}^e = \boldsymbol{T}^{\mathrm{T}}\boldsymbol{F}^e \tag{5-98}$$

将单元矩阵和向量组集后，刚架结构的运动方程为

$$\bar{\boldsymbol{M}}\ddot{\bar{\boldsymbol{\delta}}} + \bar{\boldsymbol{K}}\bar{\boldsymbol{\delta}} = \bar{\boldsymbol{F}}(t) \tag{5-99}$$

5.3 空间刚架

组成刚架结构的梁构件在几何上不处于同一平面，或结构所受的载荷不在结构的平面内，则称为空间刚架系统. 空间刚架结构除了承受拉压和弯曲载荷作用外，还可承受扭矩作用，而且弯曲变形可能同时在两个坐标平面内存在，单元的特性矩阵将是几种单元特性矩阵的组合.

空间刚架的单元为三维单元，一般承受轴向力、弯矩和扭矩的作用，单元的特性是轴力、弯曲和扭转单元特性的组合. 在小变形的情况下，空间刚架单元的特性矩阵可以由轴

力单元、弯曲单元和扭转单元的特性矩阵叠加来构成.

对于二节点单元，在单元局部坐标系下，节点变量可表示为

$$\boldsymbol{\delta}^e=\begin{bmatrix}u_1 & v_1 & w_1 & \theta_{x1} & \theta_{y1} & \theta_{z1} & u_2 & v_2 & w_2 & \theta_{x2} & \theta_{y2} & \theta_{z2}\end{bmatrix}^{\mathrm{T}} \tag{5-100}$$

式中，u_i、v_i、w_i 为节点 i 沿局部坐标方向的位移；θ_{xi}、θ_{yi}、θ_{zi} 为节点 i 处截面绕 3 个坐标轴的转角. 其中，θ_{xi} 表示截面的扭转，θ_{yi} 和 θ_{zi} 分别表示截面在 xz 和 xy 坐标平面内的转动.

节点力可表示为

$$\boldsymbol{F}^e=\begin{bmatrix}U_1 & V_1 & W_1 & M_{x1} & M_{y1} & M_{z1} & U_2 & V_2 & W_2 & M_{x2} & M_{y2} & M_{z2}\end{bmatrix}^{\mathrm{T}} \tag{5-101}$$

式中：U_i 是节点 i 的轴向力；V_i、W_i 是节点 i 在 xz 和 xy 坐标平面内的剪力；M_{xi} 是节点 i 的扭矩；M_{yi}、M_{zi} 是节点 i 在 xz 和 xy 坐标平面内的弯矩.

弯曲单元采用 Euler-Bernoulli 梁的情况下，三维刚架单元的刚度矩阵为

$$\boldsymbol{K}^e=\begin{bmatrix}
\frac{EA}{l} & 0 & 0 & 0 & 0 & 0 & -\frac{EA}{l} & 0 & 0 & 0 & 0 & 0\\
 & \frac{12EI_z}{l^3} & 0 & 0 & 0 & \frac{6EI_z}{l^2} & 0 & -\frac{12EI_z}{l^3} & 0 & 0 & 0 & \frac{6EI_z}{l^2}\\
 & & \frac{12EI_y}{l^3} & 0 & -\frac{6EI_y}{l^2} & 0 & 0 & 0 & -\frac{12EI_y}{l^3} & 0 & -\frac{6EI_y}{l^2} & 0\\
 & & & \frac{GJ}{l} & 0 & 0 & 0 & 0 & 0 & -\frac{GJ}{l} & 0 & 0\\
 & & & & \frac{4EI_y}{l} & 0 & 0 & 0 & \frac{6EI_y}{l^2} & 0 & \frac{2EI_y}{l} & 0\\
 & & & & & \frac{4EI_z}{l} & 0 & -\frac{6EI_z}{l^2} & 0 & 0 & 0 & \frac{2EI_z}{l}\\
 & & & & & & \frac{EA}{l} & 0 & 0 & 0 & 0 & 0\\
 & \text{对} & & & & & & \frac{12EI_z}{l^3} & 0 & 0 & 0 & -\frac{6EI_z}{l^2}\\
 & & & & & & & & \frac{12EI_y}{l^3} & 0 & \frac{6EI_y}{l^2} & 0\\
 & & & & & & & & & \frac{GJ}{l} & 0 & 0\\
 & & & & \text{称} & & & & & & \frac{4EI_y}{l} & 0\\
 & & & & & & & & & & & \frac{4EI_z}{l}
\end{bmatrix} \tag{5-102}$$

式中：A 为单元的横截面面积；I_y 是在 xz 坐标平面内的截面惯性矩；I_z 是在 xy 坐标平面内的截面惯性矩；J 是单元的极惯性矩.

质量矩阵为

$$
\boldsymbol{M}^{e}=\frac{\rho Al}{420}\begin{bmatrix}
140 & 0 & 0 & 0 & 0 & 0 & 70 & 0 & 0 & 0 & 0 & 0\\
 & 156 & 0 & 0 & 0 & 22l & 0 & 54 & 0 & 0 & 0 & -13l\\
 & & 156 & 0 & -22l & 0 & 0 & 0 & 54 & 0 & 13l & 0\\
 & & & \frac{140}{A}J & 0 & 0 & 0 & 0 & 0 & \frac{70}{A}J & 0 & 0\\
 & & & & 4l^2 & 0 & 0 & 0 & -13l & 0 & -3l^2 & 0\\
 & & & & & 4l^2 & 0 & 13l & 0 & 0 & 0 & -3l^2\\
 & & & & & & 140 & 0 & 0 & 0 & 0 & 0\\
 & \text{对} & & & & & & 156 & 0 & 0 & 0 & -22l\\
 & & & & & & & & 156 & 0 & 22l & 0\\
 & & & & & & & & & \frac{140}{A}J & 0 & 0\\
 & & & & \text{称} & & & & & & 4l^2 & 0\\
 & & & & & & & & & & & 4l^2
\end{bmatrix} \tag{5-103}
$$

而集中质量矩阵为

$$
\boldsymbol{M}^{e}=\frac{\rho Al}{2}\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
 & & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
 & & & \frac{J}{A} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
 & & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
 & & & & & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
 & & & & & & 1 & 0 & 0 & 0 & 0 & 0\\
 & \text{对} & & & & & & 1 & 0 & 0 & 0 & 0\\
 & & & & & & & & 1 & 0 & 0 & 0\\
 & & & & & & & & & \frac{J}{A} & 0 & 0\\
 & & & & \text{称} & & & & & & 0 & 0\\
 & & & & & & & & & & & 0
\end{bmatrix} \tag{5-104}
$$

弯曲单元采用 Timoshenko 梁的情况下，三维刚架单元的刚度矩阵为

$$
K^e=\begin{bmatrix}
\frac{EA}{l} & 0 & 0 & 0 & 0 & 0 & -\frac{EA}{l} & 0 & 0 & 0 & 0 & 0\\
 & \frac{GA}{kl} & 0 & 0 & 0 & \frac{GA}{2k} & 0 & -\frac{GA}{kl} & 0 & 0 & 0 & \frac{GA}{2k}\\
 & & \frac{GA}{kl} & 0 & -\frac{GA}{2k} & 0 & 0 & 0 & -\frac{GA}{kl} & 0 & -\frac{GA}{2k} & 0\\
 & & & \frac{GJ}{l} & 0 & 0 & 0 & 0 & 0 & -\frac{GJ}{l} & 0 & 0\\
 & & & & \frac{GAl}{4k}+\frac{EI_y}{l} & 0 & 0 & 0 & \frac{GA}{2k} & 0 & \frac{GAl}{4k}-\frac{EI_y}{l} & 0\\
 & & & & & \frac{GAl}{4k}+\frac{EI_z}{l} & 0 & -\frac{GA}{2k} & 0 & 0 & 0 & \frac{GAl}{4k}-\frac{EI_z}{l}\\
 & & & & & & \frac{EA}{l} & 0 & 0 & 0 & 0 & 0\\
 & 对 & & & & & & \frac{GA}{kl} & 0 & 0 & 0 & -\frac{GA}{2k}\\
 & & & & & & & & \frac{GA}{kl} & 0 & \frac{GA}{2k} & 0\\
 & & & & & & & & & \frac{GJ}{l} & 0 & 0\\
 & & & & 称 & & & & & & \frac{GAl}{4k}+\frac{EI_y}{l} & 0\\
 & & & & & & & & & & & \frac{GAl}{4k}+\frac{EI_z}{l}
\end{bmatrix}
\tag{5-105}
$$

质量矩阵为

$$
M^e=\frac{\rho Al}{6}\begin{bmatrix}
2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\
 & 2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\
 & & 2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\
 & & & 2\frac{J}{A} & 0 & 0 & 0 & 0 & 0 & \frac{J}{A} & 0 & 0\\
 & & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
 & & & & & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
 & & & & & & 2 & 0 & 0 & 0 & 0 & 0\\
 & 对 & & & & & & 2 & 0 & 0 & 0 & 0\\
 & & & & & & & & 2 & 0 & 0 & 0\\
 & & & & & & & & & 2\frac{J}{A} & 0 & 0\\
 & & & & 称 & & & & & & 0 & 0\\
 & & & & & & & & & & & 0
\end{bmatrix}
\tag{5-106}
$$

而集中质量矩阵与式(5-104)相同.

对于空间刚架单元，局部坐标系与总体坐标系的转换矩阵为

$$\boldsymbol{T}=\begin{bmatrix} \boldsymbol{t} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{t} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{t} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{t} \end{bmatrix} \tag{5-107}$$

其中

$$\boldsymbol{t}=\begin{bmatrix} \cos(x,\bar{x}) & \cos(x,\bar{y}) & \cos(x,\bar{z}) \\ \cos(y,\bar{x}) & \cos(y,\bar{y}) & \cos(y,\bar{z}) \\ \cos(z,\bar{x}) & \cos(z,\bar{y}) & \cos(z,\bar{z}) \end{bmatrix} \tag{5-108}$$

其中，子矩阵$\boldsymbol{t}$的第1行可由节点的坐标计算

$$\cos(x,\bar{x})=\frac{\bar{x}_2-\bar{x}_1}{l},\quad \cos(x,\bar{y})=\frac{\bar{y}_2-\bar{y}_1}{l},\quad \cos(x,\bar{z})=\frac{\bar{z}_2-\bar{z}_1}{l}$$

$$l=\sqrt{(\bar{x}_2-\bar{x}_1)^2+(\bar{y}_2-\bar{y}_1)^2+(\bar{z}_2-\bar{z}_1)^2}$$

而子矩阵$\boldsymbol{t}$其他行的确定需要用一个辅助参考坐标系$x'-y'-z'$[4]。

辅助参考坐标系$x'-y'-z'$与总体坐标系$\bar{x}-\bar{y}-\bar{x}$间的转换矩阵为

$$\boldsymbol{t}_1=\begin{bmatrix} l & m & n \\ -\dfrac{m}{\lambda} & \dfrac{l}{\lambda} & 0 \\ -\dfrac{nl}{\lambda} & -\dfrac{mn}{\lambda} & \lambda \end{bmatrix} \tag{5-109}$$

其中

$$\begin{aligned} &\cos(x,\bar{x})=\frac{(\bar{x}_j-\bar{x}_i)}{L}=l,\ \cos(x,\bar{y})=\frac{(\bar{y}_j-\bar{y}_i)}{L}=m \\ &\cos(x,\bar{z})\frac{(\bar{z}_j-\bar{z}_i)}{L}=n,\ \lambda=\sqrt{l^2+m^2} \end{aligned} \tag{5-110}$$

其中

$$L=\sqrt{(\bar{x}_j-\bar{x}_i)^2+(\bar{y}_j-\bar{y}_i)^2+(\bar{z}_j-\bar{z}_i)^2}$$

而辅助参考坐标系$x'-y'-z'$与局部坐标系$x-y-z$的转换矩阵为

$$\boldsymbol{t}_2=\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix} \tag{5-111}$$

式中，α是局部坐标系y轴相应于辅助坐标系y'轴的夹角。于是，子矩阵$\boldsymbol{t}$可表示为

$$\boldsymbol{t}=\boldsymbol{t}_2\boldsymbol{t}_1=\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix}\begin{bmatrix} l & m & n \\ -\dfrac{m}{\lambda} & \dfrac{l}{\lambda} & 0 \\ -\dfrac{nl}{\lambda} & -\dfrac{mn}{\lambda} & \lambda \end{bmatrix} \tag{5-112}$$

若单元与总体坐标系垂直，即单元的x轴平行于总体坐标系的$\bar{z}$轴，则变换矩阵$\boldsymbol{t}_1$变

为

$$t_2=\begin{bmatrix}0 & 0 & 1\\ -\sin\alpha & \cos\alpha & 0\\ -\cos\alpha & -\sin\alpha & 0\end{bmatrix} \tag{5-113}$$

因此，局部坐标系下的单元节点位移与总体坐标系中的单元节点位移的关系可写成

$$\boldsymbol{\delta}^e=\boldsymbol{T}\bar{\boldsymbol{\delta}}^e \tag{5-114}$$

将式(5-114)代入有限元的方程中，并对方程前乘$\boldsymbol{T}^{\mathrm{T}}$，即可得到转换到总体坐标系下的单元特性矩阵

$$\bar{\boldsymbol{K}}^e=\boldsymbol{T}^{\mathrm{T}}\boldsymbol{K}^e\boldsymbol{T},\quad \bar{\boldsymbol{M}}^e=\boldsymbol{T}^{\mathrm{T}}\boldsymbol{M}^e\boldsymbol{T} \tag{5-115}$$

和力向量

$$\bar{\boldsymbol{F}}^e=\boldsymbol{T}^{\mathrm{T}}\boldsymbol{F}^e \tag{5-116}$$

将单元矩阵和向量组集后，刚架结构的运动方程为

$$\bar{\boldsymbol{M}}\ddot{\bar{\boldsymbol{\delta}}}+\bar{\boldsymbol{K}}\bar{\boldsymbol{\delta}}=\bar{\boldsymbol{F}}(t) \tag{5-117}$$

5.4　应用问题与 MATLAB 程序

对于本章讨论的梁和刚架的分析，可用 MATLAB 来实现. 下面以实例来介绍基于 MATLAB 的应用问题.

5.4.1　静力学问题分析

对于梁或刚架的静力学分析，有限元方程变为

$$\boldsymbol{K}\boldsymbol{\delta}=\boldsymbol{F} \tag{5-118}$$

式中的矩阵和向量为经过组集后系统矩阵和向量，且均为总体坐标系下的表示式. 在不至于混淆的情况下，为了方便，本书后面在表示总体坐标系的矩阵和向量时将省略“¯”的符号.

【例 5.1】 对于如图 5.6 所示的简支梁，梁的长度为 1m，横截面面积为 0.02×0.02m^2，梁的弹性模量为 2.1×10^{11}Pa，质量密度为 7 860kg/m^3. 在梁的中部作用有-500N 的力，分别用 Euler Bernoulli 梁单元和 Timoshenko 梁单元计算梁的变形.

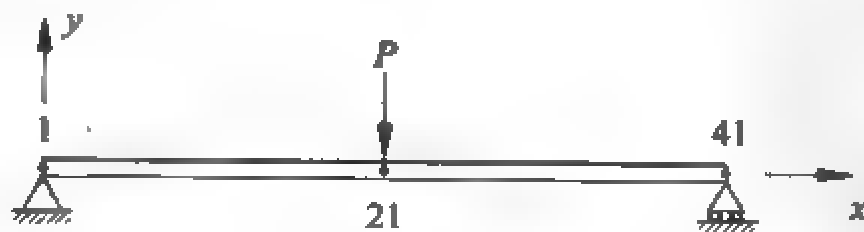


图 5.6　简支梁模型

将图 5.6 所示简支梁划分成 40 个单元进行有限元分析，用 MATLAB 编程计算，其中主要用到的 MATLAB 函数(见 5.5 节)有：

- BeamElement1(prop,leng,Opt_mass)——用 Euler Bernoulli 梁单元计算单元的刚度

矩阵 k 和质量矩阵 m.

- BeamElement2(prop,leng,Opt_mass) ——用 Timoshenko 梁单元计算单元的刚度矩阵 k 和质量矩阵 m.
- femEldof(nd,No nel,No_dof)——计算单元自由度对应到结构系统上的自由度.
- femAssemble1(kk,k,index)——用于将单元刚度矩阵或质量矩阵组集到系统的总体刚度矩阵 kk 或总体质量矩阵 mm.
- femApplybc1(kk,mm,ff,bcdof,bcval)——对结构系统施加边界条件.

M 文件如下：

```
%----------------------------------------------------------------------------
% Example 5.1
%----------------------------------------------------------------------------
%   To solve the static response for a beam structure
%   using Euler Bernoulli beam or Timoshenko beam element.
% Problem description
%   Find the deflection of a simply supported beam whose length is 1 m.
%   The beam has elastic modulus of 2.1x10^11 and its cross-section is
%   0.02 m height by 0.02 m width.
%   The mass density is 7860 kg/m^3. A concentrated load of -500 N is
%   applied at the middle of the beam.
% Variable descriptions
%   k, m - element stiffness matrix and element mass matrix
%   kk, mm - system stiffness matrix and system mass matrix
%   ff - system force vector
%   index - a vector containing system dofs associated with each element
%   bcdof - a vector containing dofs associated with boundary conditions
%   bcval - a vector containing boundary condition values associated
%   with the dofs in bcdof
%----------------------------------------------------------------------------
%  (0) input data
%----------------------------------------------------------------------------
clear; clc;

Beam_InputData541;    % import the input data for the information of
                      % nodes, elements, loads, constraints and materials
Opt_beam=1;                        % option for type of the beam:
                                   % =1 Euler Bernoulli beam
                                   % =2 Timoshenko beam
Opt_mass=2;                        % option for mass matrix:
                                   % =1 consistent mass matrix
                                   % =2 lumped mass matrix
Opt_graphics1=1;      % option for graphics of the nodal connectivity
Opt_graphics2=1;      % option for graphics of the static deformation
%----------------------------------------------------------------------------
%  (1) initialization of matrices and vectors to zero
```

```
%-----------------------------------------------------------------------
k=zeros(No_nel*No_dof,No_nel*No_dof);        % element stiffness matrix
m=zeros(No_nel*No_dof,No_nel*No_dof);        % element mass matrix

kk=zeros(Sys_dof,Sys_dof);  % initialization of system stiffness matrix
mm=zeros(Sys_dof,Sys_dof);  % initialization of system mass matrix
ff=zeros(Sys_dof,1);        % initialization of system force vector

index=zeros(No_nel*No_dof,1);       % initialization of index vector

bcdof=zeros(Sys_dof,1);             % initializing the vector bcdof
bcval=zeros(Sys_dof,1);             % initializing the vector bcval
%-----------------------------------------------------------------------
%  (2) calculation of constraints
%-----------------------------------------------------------------------
[n1,n2]=size(ConNode);
                                   % calculate the constrained dofs
for ni=1:n1
  ki=ConNode(ni,1);
  bcdof((ki-1)*No_dof+1:ki*No_dof)=ConNode(ni,2:No_dof+1);
                                   % the code for constrained dofs
  bcval((ki-1)*No_dof+1:ki*No_dof)=ConVal(ni,2:No_dof+1);
                                   % the value at constrained dofs
end
%-----------------------------------------------------------------------
%  (3) applied nodal forces
%-----------------------------------------------------------------------

ff(No_dof*(P(2)-1)+P(3))=P(1);

%-----------------------------------------------------------------------
%  (4) calculate the element matrices and assembling
%-----------------------------------------------------------------------
for iel=1:No_el                 % loop for the total number of elements
  nd(1)=iel;                    % 1st node number of the iel-th element
  nd(2)=iel+1;                  % 2nd node number of the iel-th element
  x(1)=gcoord(nd(1),1); y(1)=gcoord(nd(1),2); z(1)=gcoord(nd(1),3);
                                % coordinates of 1st node
  x(2)=gcoord(nd(2),1); y(2)=gcoord(nd(2),2); z(2)=gcoord(nd(2),3);
                                % coordinates of 2nd node
  leng=sqrt((x(2)-x(1))^2+(y(2)-y(1))^2+(z(2)-z(1))^2);
                                % length of element 'iel'
  if Opt_beam==1
    [k,m]=BeamElement1(prop,leng,Opt_mass);
             % compute element matrices for Euler-Bernoulli beam
  else
    [k,m]=BeamElement2(prop,leng,Opt_mass);
```

```
              % compute element matrices for Timoshenko beam
  end

  index=femEldof(nd,No_nel,No_dof);
  % extract system dofs associated with element

  kk=femAssemble1(kk,k,index);     % assemble system stiffness matrix
  mm=femAssemble1(mm,m,index);     % assemble system mass matrix
end
%-----------------------------------------------------------------------
%  (5) apply constraints and solve the matrix equation
%-----------------------------------------------------------------------
[kk1,mm1,ff1]=femApplybc1(kk,mm,ff,bcdof,bcval);
                   % apply boundary conditions to the system equation
displmt=kk1\ff1;
                                  % solve the matrix equation
di=Sys_dof/No_dof;
for ii=1:di
  for jj=1:No_dof
    displmtnode(ii,jj)=displmt(No_dof*(ii-1)+jj,1);
  end
end
%-----------------------------------------------------------------------
%  (6) Analytical solution
%-----------------------------------------------------------------------
E=prop(1); Iz=prop(8); L=1;
nk=ceil(No_node/2);

for ii=1:nk
  Lx=(ii-1)*dx;
  c=P(1)/(48*E*Iz);
  Asolution(ii,1)=c*(3*L^2-4*Lx^2)*Lx;
  Asolution(ii,2)=c*(3*L^2-12*Lx^2);
end
for ii=nk+1:No_node
  Lx=(ii-1)*dx;
  c=P(1)/(48*E*Iz);
  Asolution(ii,1)=c*((3*L^2-4*Lx^2)*Lx+(2*Lx-L)^3);
  Asolution(ii,2)=c*((3*L^2-12*Lx^2)+6*(2*Lx-L)^2);
end
%-----------------------------------------------------------------------
%  (7) graphics of nodal connectivity and static deformation
%-----------------------------------------------------------------------
%---------------------------------
%  (7.1) display the nodal connectivity
%---------------------------------
if Opt_graphics1==1
```

```
  for iel=1:No_el                 % loop for the total number of elements
    nd(1)=iel;                    % 1st node number of the iel-th element
    nd(2)=iel+1;                  % 2nd node number of the iel-th element
    x(1)=gcoord(nd(1),1); y(1)=gcoord(nd(1),2); z(1)=gcoord(nd(1),3);
                                            % coordinates of 1st node
    x(2)=gcoord(nd(2),1); y(2)=gcoord(nd(2),2); z(2)=gcoord(nd(2),3);
                                            % coordinates of 2nd node
    figure(1)
      plot(x,y), xlabel('x'), ylabel('y'), hold on;
      axis([-0.1,1.1,-0.01,0.01]);
  end
  if Opt_graphics2~=1
    title('nodal connectivity of elements'), hold off;
  end
end
%----------------------------------
%  (7.2) display the static deformation
%----------------------------------
if Opt_graphics2==1
  gcoordA=gcoord+[zeros(No_node,1),displmtnode(:,1),zeros(No_node,1)];
  for iel=1:No_el                 % loop for the total number of elements
    nd(1)=iel;                    % 1st node number of the iel-th element
    nd(2)=iel+1;                  % 2nd node number of the iel-th element
    x(1)=gcoordA(nd(1),1); y(1)=gcoordA(nd(1),2); z(1)=gcoordA(nd(1),3);
                                            % coordinates of 1st node
    x(2)=gcoordA(nd(2),1); y(2)=gcoordA(nd(2),2); z(2)=gcoordA(nd(2),3);
                                            % coordinates of 2nd node
    figure(1)
      plot(x,y,'r'), title('Deflection of the structure'), hold on;
  end
  hold off
end
%----------------------------------------------------------------------
%  (8) print fem solutions
%----------------------------------------------------------------------
disp('The calculation is use of:')
if Opt_beam==1
  disp('Euler-Bernoulli beam')
else
  disp('Timoshenko beam element')
end
disp(' ')
disp('                 numrical          analytical')
disp('     node        y        θ        y        θ')
num=1:di;
```

```
displacements=[num' displmtnode Asolution]   % print nodal displacements
%-----------------------------------------------------------------------
%    The end
% ----------------------------------------------------------------------
```

```
% ----------------------------------------------------------------------
% Input data for static analysis of beam structure (Example 5.4.1)
% ----------------------------------------------------------------------
%  (0.1) input basic data
% ----------------------------------------------------------------------
Prob title='static analysis of beam structure';

No_el=40;                              % number of elements
No_nel=2;                              % number of nodes per element
No_dof=2;                              % number of dofs per node
No_node=(No_nel-1)*No_el+1;           % total number of nodes in system
Sys_dof=No_node*No_dof;                % total system dofs
%-----------------------------------------------------------------------
%  (0.2) physical and geometric properties
%-----------------------------------------------------------------------
prop(1)=2.1e11;                                          % elastic modulus
prop(2)=0.3;                                              % Poisson's ratio
prop(3)=7860;             % mass density (mass per unit volume)
prop(4)=0.02;             % height of beam cross-section in y direction
prop(5)=0.02;             % width of beam cross-section in z direction
prop(6)=0.02*0.02;        % cross-sectional area of the beam
prop(7)=0.02*0.02^3/12;   % moment of inertia of cross-section about axis y
prop(8)=0.02*0.02^3/12;   % moment of inertia of cross-section about axis z
%-----------------------------------------------------------------------
%  (0.3) nodal coordinates
%-----------------------------------------------------------------------
                       % x, y, z coordinates in global coordinate system
xx=zeros(No_node,1); yy=zeros(No_node,1); zz=zeros(No_node,1);
dx=0.025;
xx=0:dx:(No_node-1)*dx; xx=xx';
gcoord=[xx  yy  zz];
%-----------------------------------------------------------------------
%  (0.4) applied load
%-----------------------------------------------------------------------

P=[-500,21,1];   % load applied at node 21 in the negative y direction

%-----------------------------------------------------------------------
%  (0.5) boundary conditions
%-----------------------------------------------------------------------
ConNode=[ 1, 1, 0;...                 % code for constrained nodes
```

```
        41, 1, 0];
ConVal =[ 1, 0, 0;...                          % values at constrained dofs
        41, 0, 0];
%-----------------------------------------------------------------------
%    The end
%-----------------------------------------------------------------------
```

分别用 Euler Bernoulli 梁单元和 Timoshenko 梁单元计算梁的变形，并与解析解比较，见表 5.1.

表 5.1　有限元解与解析解的比较

节　点	有限元解		解析解	
	v	θ	v	θ
1	0	−0.0112	0	−0.0112
2	−0.0003	−0.0111	−0.0003	−0.0111
3	−0.0006	−0.0110	−0.0006	−0.0110
4	−0.0008	−0.0109	−0.0008	−0.0109
5	−0.0011	−0.0107	−0.0011	−0.0107
6	−0.0014	−0.0105	−0.0014	−0.0105
7	−0.0016	−0.0102	−0.0016	−0.0102
8	−0.0019	−0.0098	−0.0019	−0.0098
9	−0.0021	−0.0094	−0.0021	−0.0094
10	−0.0023	−0.0089	−0.0023	−0.0089
11	−0.0026	−0.0084	−0.0026	−0.0084
12	−0.0028	−0.0078	−0.0028	−0.0078
13	−0.0029	−0.0071	−0.0029	−0.0071
14	−0.0031	−0.0064	−0.0031	−0.0064
15	−0.0033	−0.0057	−0.0033	−0.0057
16	−0.0034	−0.0049	−0.0034	−0.0049
17	−0.0035	−0.0040	−0.0035	−0.0040
18	−0.0036	−0.0031	−0.0036	−0.0031
19	−0.0037	−0.0021	−0.0037	−0.0021
20	−0.0037	−0.0011	−0.0037	−0.0011
21	−0.0037	0.0000	−0.0037	0
⋮	⋮	⋮	⋮	⋮
41	0	0.0112	0	0.0112

【例 5.2】 对于如图 5.6 所示的简支梁，用混合梁单元计算梁的变形.

M 文件如下:

```
%------------------------------------------------------------------------
% Example 5.2
%   To solve the static response for a beam structure using mixed beam elements.
%   nodal dofs: {M1  v1  M2  v2}
% Problem description
%   Find the deflection of a simply supported beam described in Example 5.4.1.
%------------------------------------------------------------------------
%  (0) input data
%------------------------------------------------------------------------
clear; clc;

Beam_InputData542;   % import the input data for the information
                     % of nodes, elements, loads, constraints and materials
Opt_mass=2;          % option for mass matrix:
                     % =1 consistent mass matrix
                     % =2 lumped mass matrix
Opt_graphics1=1;     % option for graphics of the nodal connectivity
Opt_graphics2=1;     % option for graphics of the static deformation
%------------------------------------------------------------------------
%  (1) initialization of matrices and vectors to zero
%------------------------------------------------------------------------
k=zeros(No_nel*No_dof,No_nel*No_dof);         % element stiffness matrix
m=zeros(No_nel*No_dof,No_nel*No_dof);         % element mass matrix

kk=zeros(Sys_dof,Sys_dof);  % initialization of system stiffness matrix
mm=zeros(Sys_dof,Sys_dof);  % initialization of system mass matrix
ff=zeros(Sys_dof,1);        % initialization of system force vector

index=zeros(No_nel*No_dof,1);       % initialization of index vector

bcdof=zeros(Sys_dof,1);             % initializing the vector bcdof
bcval=zeros(Sys_dof,1);             % initializing the vector bcval
%------------------------------------------------------------------------
%  (2) calculation of constraints
%------------------------------------------------------------------------
[n1,n2]=size(ConNode);
                                    % calculate the constrained dofs
for ni=1:n1
  ki=ConNode(ni,1);
  bcdof((ki-1)*No_dof+1:ki*No_dof)=ConNode(ni,2:No_dof+1);
                                    % the code for constrained dofs
  bcval((ki-1)*No_dof+1:ki*No_dof)=ConVal(ni,2:No_dof+1);
                                    % the value at constrained dofs
end
%------------------------------------------------------------------------
%  (3) applied nodal forces
%------------------------------------------------------------------------
```

```
ff(No_dof*(P(2)-1)+P(3))=-P(1);

%-----------------------------------------------------------------------
%  (4) calculate the element matrices and assembling
%-----------------------------------------------------------------------
for iel=1:No_el              % loop for the total number of elements
  nd(1)=iel;                 % 1st node number of the iel-th element
  nd(2)=iel+1;               % 2nd node number of the iel-th element
  x(1)=gcoord(nd(1),1); y(1)=gcoord(nd(1),2); z(1)=gcoord(nd(1),3);
                                          % coordinates of 1st node
  x(2)=gcoord(nd(2),1); y(2)=gcoord(nd(2),2); z(2)=gcoord(nd(2),3);
                                          % coordinates of 2nd node
  leng=sqrt((x(2)-x(1))^2+(y(2)-y(1))^2+(z(2)-z(1))^2);
                                          % length of element 'iel'
  [k,m]=BeamElement4(prop,leng,Opt_mass);
                             % compute element matrices for mixed beam

  index=femEldof(nd,No_nel,No_dof);
  % extract system dofs associated with element

  kk=femAssemble1(kk,k,index);      % assemble system stiffness matrix
  mm=femAssemble1(mm,m,index);      % assemble system mass matrix
end
%-----------------------------------------------------------------------
%  (5) apply constraints and solve the matrix equation
%-----------------------------------------------------------------------
[kk1,mm1,ff1]=femApplybc1(kk,mm,ff,bcdof,bcval);
                        % apply boundary conditions to the system equation
displmt=kk1\ff1;        % solve the matrix equation

di=Sys_dof/No_dof;
for ii=1:di
  for jj=1:No_dof
    displmtnode(ii,jj)=displmt(No_dof*(ii-1)+jj,1);
  end
end
%-----------------------------------------------------------------------
%  (6) Analytical solution
%-----------------------------------------------------------------------
E=prop(1); Iz=prop(8); L=1;
nk=ceil(No_node/2);

for ii=1:nk
  Lx=(ii-1)*dx;
  c=P(1)/(48*E*Iz);
  Asolution(ii,1)=250*Lx;
  Asolution(ii,2)=c*(3*L^2-4*Lx^2)*Lx;
end
```

```
for ii=nk+1:No_node
  Lx=(ii-1)*dx;
  c=P(1)/(48*E*Iz);
  Asolution(ii,1)=250*Lx-500*(Lx-L/2);
  Asolution(ii,2)=c*((3*L^2-4*Lx^2)*Lx+(2*Lx-L)^3);
end
%--------------------------------------------------------------------------
%  (7) graphics of nodal connectivity and static deformation
%--------------------------------------------------------------------------
%---------------------------------
%  (7.1) display the nodal connectivity
%---------------------------------
if Opt_graphics1==1
  for iel=1:No_el                 % loop for the total number of elements
    nd(1)=iel;                   % 1st node number of the iel-th element
    nd(2)=iel+1;                 % 2nd node number of the iel-th element
    x(1)=gcoord(nd(1),1); y(1)=gcoord(nd(1),2); z(1)=gcoord(nd(1),3);
                                          % coordinates of 1st node
    x(2)=gcoord(nd(2),1); y(2)=gcoord(nd(2),2); z(2)=gcoord(nd(2),3);
                                          % coordinates of 2nd node
    figure(1)
      plot(x,y), xlabel('x'), ylabel('y'), hold on;
      axis([-0.1,1.1,-0.01,0.01])
  end
  if Opt_graphics2~=1
    title('nodal connectivity of elements'), hold off
  end
end
%---------------------------------
%  (7.2) display the static deformation
%---------------------------------
if Opt_graphics2==1
  gcoordA=gcoord+[zeros(No_node,1),displmtnode(:,2),zeros(No_node,1)];
  for iel=1:No_el               % loop for the total number of elements
    nd(1)=iel;                  % 1st node number of the iel-th element
    nd(2)=iel+1;                % 2nd node number of the iel-th element
    x(1)=gcoordA(nd(1),1); y(1)=gcoordA(nd(1),2); z(1)=gcoordA(nd(1),3);
                                            % coordinates of 1st node
    x(2)=gcoordA(nd(2),1); y(2)=gcoordA(nd(2),2); z(2)=gcoordA(nd(2),3);
                                            % coordinates of 2nd node
    figure(1)
      plot(x,y,'r'), title('Deflection of the structure'), hold on;
  end
  hold off
end
%--------------------------------------------------------------------------
%  (8) print fem solutions
%--------------------------------------------------------------------------
disp('The calculation is use of:')
```

```
if Opt beam==1
  disp('Euler-Bernoulli beam')
else
  disp('Timoshenko beam element')
end
disp(' ')
disp('                  numrical          analytical')
disp('    node      M         y         M        y')
num=1:di;
displacements=[num' displmtnode Asolution]     % print nodal
displacements
%----------------------------------------------------------------------
%    The end
% ---------------------------------------------------------------------



% ---------------------------------------------------------------------
% Input data for static analysis of beam structure (5.2)
% ---------------------------------------------------------------------
%  (0.1) input basic data
% ---------------------------------------------------------------------
Prob_title='static analysis of beam structure';

No_el=40;                          % number of elements
No_nel=2;                          % number of nodes per element
No_dof=2;                          % number of dofs per node
No_node=(No_nel-1)*No_el+1;       % total number of nodes in system
Sys_dof=No_node*No_dof;            % total system dofs
%----------------------------------------------------------------------
%  (0.2) physical and geometric properties
%----------------------------------------------------------------------
prop(1)=2.1e11;            % elastic modulus
prop(2)=0.3;               % Poisson's ratio
prop(3)=7860;              % mass density (mass per unit volume)
prop(4)=0.02;              % height of beam cross-section in y direction
prop(5)=0.02;              % width of beam cross-section in z direction
prop(6)=0.02*0.02;         % cross-sectional area of the beam
prop(7)=0.02*0.02^3/12;    % moment of inertia of cross-section about axis y
prop(8)=0.02*0.02^3/12;    % moment of inertia of cross-section about axis z
prop(9)=0;                 % polar moment of inertia
prop(10)=0;                % shear modulus
                           % = 0 not include the shear deformation
                          % = 1 shear modulus is calculated by E/(2*(1+u))
                          % = the value for the shear deformation
%----------------------------------------------------------------------
%  (0.3) nodal coordinates
%----------------------------------------------------------------------
                        % x, y, z coordinates in global coordinate system
xx=zeros(No_node,1); yy=zeros(No_node,1); zz=zeros(No_node,1);
```

```
dx=0.025;
xx=0:dx:(No_node-1)*dx; xx=xx';
gcoord=[xx  yy  zz];
%-------------------------------------------------------------------
%  (0.4) applied load
%-------------------------------------------------------------------

P=[-500,21,2];      % load applied at node 26 in the negative y direction

%-------------------------------------------------------------------
%  (0.5) boundary conditions
%-------------------------------------------------------------------
ConNode=[ 1,  1,  1;...              % code for constrained nodes
        41,  1,  1];
ConVal =[ 1,  0,  0;...              % values at constrained dofs
        41,  0,  0];
%-------------------------------------------------------------------
%    The end
% ------------------------------------------------------------------
```

用混合梁单元计算梁的变形，并与解析解比较，见表 5.2.

表 5.2　用混合梁单元计算梁的变形时有限元解与解析解的比较

节　点	有限元解		解　析　解	
	M	v	M	v
1	0	0	0	0
2	6.2500	−0.0003	6.2500	−0.0003
3	12.5000	−0.0006	12.5000	−0.0006
4	18.7500	−0.0008	18.7500	−0.0008
5	25.0000	−0.0011	25.0000	−0.0011
6	31.2500	−0.0014	31.2500	−0.0014
7	37.5000	−0.0016	37.5000	−0.0016
8	43.7500	−0.0019	43.7500	−0.0019
9	50.0000	−0.0021	50.0000	−0.0021
10	56.2500	−0.0023	56.2500	−0.0023
11	62.5000	−0.0026	62.5000	−0.0026
12	68.7500	−0.0028	68.7500	−0.0028
13	75.0000	−0.0029	75.0000	−0.0029
14	81.2500	−0.0031	81.2500	−0.0031
15	87.5000	−0.0033	87.5000	−0.0033
16	93.7500	−0.0034	93.7500	0.0034
17	100.0000	−0.0035	100.0000	0.0035

续表

节　点	有限元解		解 析 解	
	M	v	M	v
18	106.2500	−0.0036	106.2500	−0.0036
19	112.5000	−0.0037	112.5000	−0.0037
20	118.7500	−0.0037	118.7500	−0.0037
21	125.0000	−0.0037	125.0000	−0.0037
⋮	⋮	⋮	⋮	⋮
41	0	0	0	0

【例 5.3】 如图 5.7 所示为一平面刚架结构. 梁的横截面面积为 $0.05\times0.05\ \text{m}^2$，梁的弹性模量为 2.1×10^{11}Pa，质量密度为 7 860 kg/m^3. 在刚架的上部作用 P=500N 的横向载荷和 M=500 N • m 的力矩，试分析该刚架的变形.

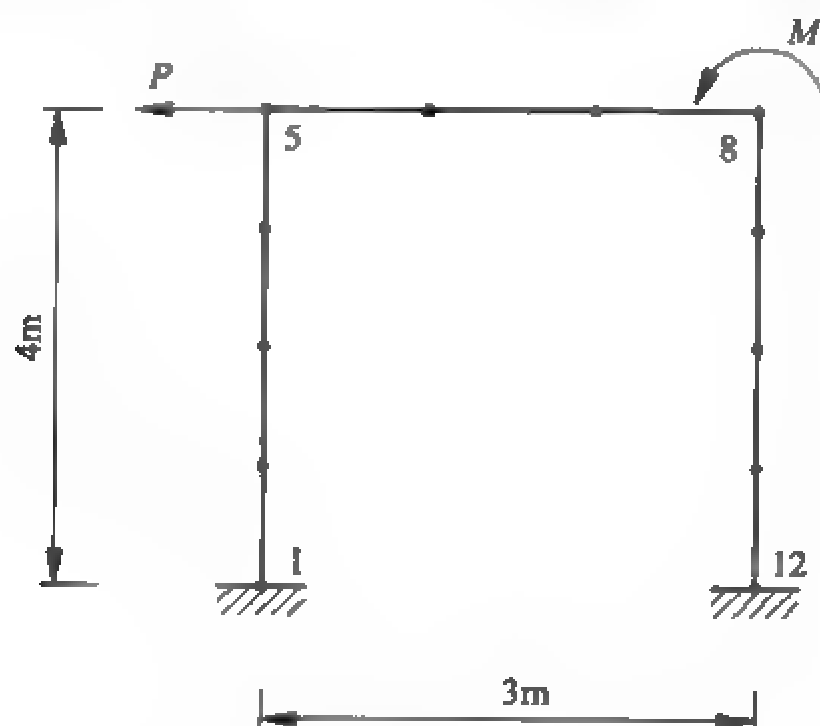


图 5.7　平面刚架模型

将刚架按图 5.7 划分成 11 个单元进行有限元分析，用 MATLAB 编程计算，其中主要用到的 MATLAB 的函数(见 5.5 节)有：

- FrameElement21(prop,leng,beta,Opt_section,Opt_mass)——用 Euler Bernoulli 梁计算单元的刚度矩阵 k 和质量矩阵 m.
- FrameElement22(prop,leng,beta,Opt_section,Opt_mass) ——用 Timoshenko 梁计算单元的刚度矩阵 k 和质量矩阵 m.
- femEldof(nd,No_nel,No_dof)——计算单元自由度对应到结构系统上的自由度.
- femAssemble1(kk,k,index)——用于将单元刚度矩阵或质量矩阵组集到系统的总体刚度矩阵 kk 或总体质量矩阵 mm.
- femApplybc1(kk,mm,ff,bcdof,bcval)——对结构系统施加边界条件.

M 文件如下：

```
%--------------------------------------------------------------------------
%
% Example 5.3
%   To solve a static deflection of a 2-D frame structure
%   nodal dofs: {u1  v1  θ1  u2  v2  θ2}
% Problem description
%   Find the deflection of a frame structure which is made of three beams
%   of lengths of 4 m, 3 m and 4 m, respectively. All beams have cross-
%   sections of 0.05 m height by 0.05 m width. The elastic modulus is
%   2.10x10^11 Pa. The frame is subjected to a concentrated load of 500 N
%   and a moment of 500 Nm at the ends of the top beam. One end of the
%   each vertical beam is fixed.
%   (see Fig. 5-9 for the element discretization)
% Variable descriptions
%   k,m = element stiffness matrix and element mass matrix
%   kk,mm = system stiffness matrix and system mass matrix
%   ff = system force vector
%   index = a vector containing system dofs associated with each element
%   bcdof = a vector containing dofs associated with boundary conditions
%   bcval = a vector containing boundary condition values associated with
%           the dofs in 'bcdof'
%   displmt = nodal displacement vector
%--------------------------------------------------------------------------
%  (0) input control data
%--------------------------------------------------------------------------
clear; clc;

Beam InputData543;     % import the input data for the information of
                       % nodes, elements, loads, constraints and materials
Opt_beam=1;                                 % option for type of the beam
                                            % =1 Euler Bernoulli beam
                                            % =2 Timoshenko beam
Opt_mass=2;                                 % option for mass matrix
                                            % =1 consistent mass matrix
                                            % =2 lumped mass matrix
Opt_section=1;                              % option for type of cross-section
                                            % = 1 rectangular cross-section
                                            % = 2 circular cross-section
Opt_graphics1=1;        % option for graphics of the nodal connectivity
Opt_graphics2=1;        % option for graphics of the static deformation
%--------------------------------------------------------------------------
%  (1) initialization of matrices and vectors to zero
%--------------------------------------------------------------------------
k=zeros(No_nel*No dof,No_nel*No_dof);        % element stiffness matrix
m=zeros(No_nel*No_dof,No_nel*No_dof);        % element mass matrix
```

```
kk=zeros(Sys_dof,Sys_dof);   % initialization of system stiffness matrix
mm=zeros(Sys_dof,Sys_dof);   % initialization of system mass matrix
ff=zeros(Sys_dof,1);         % initialization of system force vector

bcdof=zeros(Sys_dof,1);      % initializing the vector bcdof
bcval=zeros(Sys_dof,1);      % initializing the vector bcval

index=zeros(No_nel*No_dof,1);      % initialization of index vector
%-------------------------------------------------------------------------
%  (2) calculation of constraints
%-------------------------------------------------------------------------
[n1,n2]=size(ConNode);
                                   % calculate the constrained dofs
for ni=1:n1
  ki=ConNode(ni,1);
  bcdof((ki-1)*No_dof+1:ki*No_dof)=ConNode(ni,2:No_dof+1);
                                   % the code for constrained dofs
  bcval((ki-1)*No_dof+1:ki*No_dof)=ConVal(ni,2:No_dof+1);
                                   % the value at constrained dofs
end
%-------------------------------------------------------------------------
%  (3) applied nodal loads
%-------------------------------------------------------------------------
[n1,n2]=size(P);

for ni=1:n1
  ff(No_dof*(P(ni,2)-1)+P(ni,3))=P(ni,1);
end
%-------------------------------------------------------------------------
%  (4) calculate the element matrices and assembling
%-------------------------------------------------------------------------
for iel=1:No_el             % loop for the total number of elements
  nd(1)=nodes(iel,1);       % 1st connected node for the (iel)-th element
  nd(2)=nodes(iel,2);       % 2nd connected node for the (iel)-th element
  x(1)=gcoord(nd(1),1); y(1)=gcoord(nd(1),2); z(1)=gcoord(nd(1),3);
                                          % coordinate of 1st node
  x(2)=gcoord(nd(2),1); y(2)=gcoord(nd(2),2); z(2)=gcoord(nd(2),3);
                                          % coordinate of 2nd node
  leng=sqrt((x(2)-x(1))^2+(y(2)-y(1))^2+(z(2)-z(1))^2);
                                          % length of element 'iel'
                  % compute the angle between the local and global axes
  if (x(2)-x(1))==0;
    beta=pi/2;
  elseif ((y(2)-y(1))==0)&((x(2)-x(1))<0)
    beta=pi;
  else
```

```
    beta=atan((y(2)-y(1))/(x(2)-x(1)));
  end

  if Opt beam==1
    [k,m]=FrameElement21(prop,leng,beta,Opt section,Opt_mass);
                        % compute element matrix for Euler-Bernoulli beam
  elseif Opt beam==2
    [k,m]=FrameElement22(prop,leng,beta,Opt_section,Opt_mass);
                        % compute element matrix for Timoshenko beam
  end

  index=femEldof(nd,No_nel,No_dof);
                        % extract system dofs associated with element
  kk=femAssemble1(kk,k,index);      % assemble system stiffness matrix
  mm=femAssemble1(mm,m,index);      % assemble system mass matrix
end
%-----------------------------------------------------------------------------
%  (5) apply constraints and solve the matrix equation
%-----------------------------------------------------------------------------
[kk1,mm1,ff1,sdof1]=femApplybc1(kk,mm,ff,bcdof,bcval);
                  % apply fixed boundary conditions to the system equation
displmt=kk1\ff1;  % solve the matrix equation to find nodal displacements

No_dof1=No_dof/(Sys_dof/sdof1);
for ii=1:No_node
  for ij=1:No_dof1
    displmtnode(ii,ij)=displmt((ii-1)*No_dof1+ij,1);
  end
end
%-----------------------------------------------------------------------------
%  (6) graphics of nodal connectivity and static deformation
%-----------------------------------------------------------------------------
%----------------------------------
%  (6.1) display the nodal connectivity
%----------------------------------
if Opt_graphics1==1
  for iel=1:No_el           % loop for the total number of elements
    nd(1)=nodes(iel,1);   % 1st connected node for the (iel)-th element
    nd(2)=nodes(iel,2);   % 2nd connected node for the (iel)-th element
    x(1)=gcoord(nd(1),1); y(1)=gcoord(nd(1),2); z(1)=gcoord(nd(1),3);
                                         % coordinate of 1st node
    x(2)=gcoord(nd(2),1); y(2)=gcoord(nd(2),2); z(2)=gcoord(nd(2),3);
                                         % coordinate of 2nd node
    figure(1)
      plot(x,y), xlabel('x'), ylabel('y'), hold on;
      axis([-1.5,4.5,-1,5]);
  end
```

```
  if Opt graphics2~=1
    title('nodal connectivity of elements'), hold off;
  end
end
%--------------------------------
%  (6.2) display the static deformation
%--------------------------------
if Opt_graphics2==1
  Ampl=10;
  gcoordA=gcoord+[Ampl*displmtnode(:,1:2),zeros(No_node,1)];
  for iel=1:No_el            % loop for the total number of elements
    nd(1)=nodes(iel,1);     % 1st connected node for the (iel)-th element
    nd(2)=nodes(iel,2);     % 2nd connected node for the (iel)-th element
    x(1)=gcoordA(nd(1),1); y(1)=gcoordA(nd(1),2); z(1)=gcoordA(nd(1),3);
                            % coordinate of 1st node
    x(2)=gcoordA(nd(2),1); y(2)=gcoordA(nd(2),2); z(2)=gcoordA(nd(2),3);
                            % coordinate of 2nd node
    figure(1)
      plot(x,y,'r'), title('Deflection of the structure'), hold on;
  end
  hold off
end
%--------------------------------------------------------------------
%  (7) print fem solutions
%--------------------------------------------------------------------
disp('The calculation is use of')

if Opt_beam==1
  disp('Euler-Bernoulli beam element')
elseif Opt_beam==2
  disp('Timoshenko beam element')
end
disp(' ')
disp('                  numrical solution')
disp('     node       x       y        θ')
num=1:1:No_node;
displmts=[num' displmtnode]              % print nodal displacements
%--------------------------------------------------------------------
%     The end
%--------------------------------------------------------------------


%--------------------------------------------------------------------
% Input data for static analysis of frame structure (5.4.3)
%--------------------------------------------------------------------
%  (0.1) input basic data
```

```
% --------  -------------------------------------------------------
Prob_title='static analysis of a 2-D frame structure';

No_el=11;                              % number of elements
No_nel=2;                              % number of nodes per element
No_dof=3;                              % number of dofs per node
No_node=12;                            % total number of nodes in system
Sys_dof=No_node*No_dof;                % total system dofs
%-----------------------------------------------------------------------
%  (0.2) material and geometric properties
%-----------------------------------------------------------------------
prop(1)=2.1e11;                                          % elastic modulus
prop(2)=0.3;                                             % Poisson's ratio
prop(3)=7860;               % mass density (mass per init volume)
prop(4)=0.05;               % height of beam cross-section in y direction
prop(5)=0.05;               % width of beam cross-section in z direction
prop(6)=0.05*0.05;          % cross-sectional area of the beam
prop(7)=0.05*0.05^3/12;     % moment of inertia of cross-section about axis y
prop(8)=0.05*0.05^3/12;     % moment of inertia of cross-section about axis z

prop(9)=1;                            % polar moment of inertia
     % = 0 not include the torsion deformation
     % = 1 polar moment of inertia calculated by h*b^3 or pi*(D^4-d^4)/32
     % = a value for the torsion deformation
prop(10)=1;           % shear modulus
                      % = 0 not include the shear deformation
                      % = 1 shear modulus calculated by E/(2*(1+u))
                      % = the value for the shear deformation
Opt_section=1;        % option for correction factor for shear energy
                      % = 1 rectangular cross-section
                      % = 2 circular cross-section
%-----------------------------------------------------------------------
%  (0.3) nodal coordinates
%-----------------------------------------------------------------------
                      % x, y, z coordinates in global coordinate system
gcoord=[ 1    0.0     0.0     0.0;
       2    0.0     1.0     0.0;
       3    0.0     2.0     0.0;
       4    0.0     3.0     0.0;
       5    0.0     4.0     0.0;
       6    1.0     4.0     0.0;
       7    2.0     4.0     0.0;
       8    3.0     4.0     0.0;
       9    3.0     3.0     0.0;
      10    3.0     2.0     0.0;
      11    3.0     1.0     0.0;
      12    3.0     0.0     0.0];
```

```
gcoord=[gcoord(:,2:4)];
%--------------------------------------------------------------------------
%  (0.4) nodal connectivity of the elements
%--------------------------------------------------------------------------
nodes=[  1         1          2;
        2         2         3;
        3         3         4;
        4         4         5;
        5         5         6;
        6         6         7;
        7         7         8;
        8         8         9;
        9         9        10;
       10        10        11;
       11        11        12];
nodes=[nodes(:,2:3)];
%--------------------------------------------------------------------------
%  (0.5) applied loads
%--------------------------------------------------------------------------
       % a load applied at node 5 in -x direction and a moment at node 8
P=[-500,5,1;...
   -500,8,3];
%--------------------------------------------------------------------------
%  (0.6) boundary conditions
%--------------------------------------------------------------------------
ConNode=[ 1,  1,  1,  1;...          % code for constrained nodes
         12,  1,  1,  1];
ConVal =[ 1,  0,  0,  0;...          % values at constrained dofs
         12,  0,  0,  0];
%--------------------------------------------------------------------------
%    The end
%--------------------------------------------------------------------------
```

分别采用 Euler Bernoulli 梁和 Timoshenko 梁的刚架单元计算刚架的变形，结果见表 5.3.

表 5.3　刚架变形计算结果

节点	Euler Bernoulli 梁			Timoshenko 梁		
	u	v	θ	u	v	θ
1	0	0	0	0	0	0
2	-0.0030	0.0000	0.0055	-0.0028	0.0000	0.0055
3	-0.0102	0.0000	0.0085	-0.0098	0.0000	0.0085
4	-0.0191	0.0000	0.0089	0.0185	0.0000	0.0089
5	-0.0272	0.0000	0.0067	-0.0264	0.0000	0.0068
6	-0.0272	0.0048	0.0027	-0.0264	0.0048	0.0028

续表

节点	Euler Bernoulli 梁			Timoshenko 梁		
	u	v	θ	u	v	θ
7	0.0272	0.0052	0.0022	−0.0264	0.0052	−0.0021
8	0.0272	−0.0000	0.0082	0.0264	−0.0000	0.0082
9	0.0183	−0.0000	0.0092	−0.0177	−0.0000	0.0091
10	−0.0095	0.0000	0.0081	−0.0091	−0.0000	0.0081
11	−0.0027	−0.0000	0.0051	0.0025	−0.0000	−0.0051
12	0	0	0	0	0	0

【例 5.4】 如图 5.8 所示为一空间刚架结构. 该刚架由 12 个梁组成，各个梁的横截面面积为 $0.01\times0.01\text{m}^2$，梁的弹性模量为 2.1×10^{11}Pa，质量密度为 7 860 kg/m^3. 在刚架的上部 7 节点上沿 y 方向作用有 10 000N 的横向载荷，计算该刚架的变形.

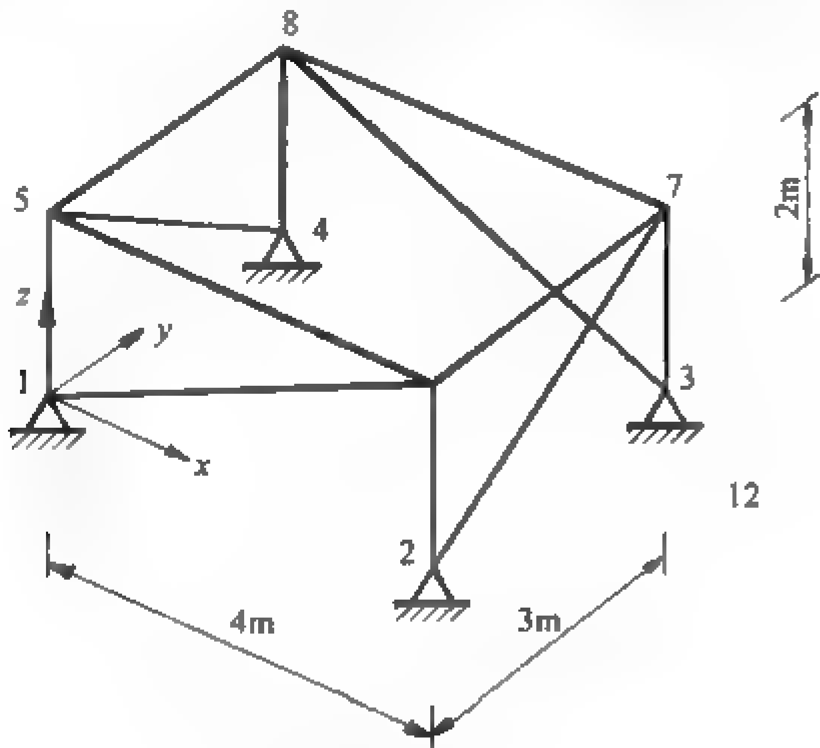


图 5.8 空间刚架模型

将刚架按图 5.8 划分成 12 个单元进行有限元分析，用 MATLAB 编程计算，其中主要用到的 MATLAB 函数(见 5.5 节)有：

- FrameElement31(prop,leng,beta,xi,al,Opt_section,Opt mass)——用 Euler Bernoulli 梁计算单元的刚度矩阵 k 和质量矩阵 m.
- FrameElement32(prop,leng,beta,xi,al,Opt_section,Opt_mass) ——用 Timoshenko 梁计算单元的刚度矩阵 k 和质量矩阵 m.
- femEldof(nd,No_nel,No_dof)——计算单元自由度对应到结构系统上的自由度.
- femAssemble1(kk,k,index)——用于将单元刚度矩阵或质量矩阵组集到系统的总体刚度矩阵 kk 或总体质量矩阵 mm.

- femApplybc1(kk,mm,ff,bcdof,bcval)——对结构系统施加边界条件.
- kkCheck1(kk,mm,ff,bcdof,bcval)——检查总体刚度矩阵 kk 的主元是否为零.

M 文件如下：

```
%----------------------------------------------------------------------%
% Example 5.4
%  To solve a static deflection of a space frame structure.
%  nodal dof: {u1  v1  w1 θ_x1 θ_y1 θ_z1  u2  v2  w2 θ_x2 θ_y2 θ_z2}
% Problem description
%  Find the static deflection of a space frame structure as shown in
%  Fig.5-11, which is made of twelve beam members. The frame are of the
%  length of 4 m, the width of 3 m and the height of 2 m. All members
%  have cross-section of 0.01 m height by 0.01 m width. The elastic modulus
%  of the structure material is 2.1e11 Pa. The mass density is 7860 kg/m^3.
%  The frame is subjected to a concentrated load of 10000N at node 7 in
%   -y direction.
% Variable descriptions
%  k,m = element stiffness matrix and element mass matrix
%  kk,mm = system stiffness matrix and system mass matrix
%  ff = system force vector
%  index = a vector containing system dofs associated with each element
%  bcdof = a vector containing dofs associated with boundary conditions
%  bcval = a vector containing boundary condition values associated
%  with the dofs in 'bcdof'
%  displmt = nodal displacement vector
%----------------------------------------------------------------------%
%  (0) input control data
%----------------------------------------------------------------------
clear; clc;

Beam_InputData544;  % import the input data for the information of
                    % nodes, elements, loads, constraints and materials
Opt_beam=1;                    % option for type of the beam:
                               % =1 Euler Bernoulli beam
                               % =2 Timoshenko beam
Opt_mass=2;                    % option for mass matrix:
                               % =1 consistent mass matrix
                               % =2 lumped mass matrix
Opt_section=1;                 % option for type of cross-section
                               % = 1 rectangular cross-section
                               % = 2 circular cross-section
Opt_graphics1=1;        % option for graphics of the nodal connectivity
Opt_graphics2=1;        % option for graphics of the static deformation
%----------------------------------------------------------------------
%  (1) initialization of matrices and vectors to zero
%----------------------------------------------------------------------
k=zeros(No_nel*No_dof,No_nel*No_dof);       % element stiffness matrix
```

```
m=zeros(No_nel*No_dof,No_nel*No_dof);      % element mass matrix

kk=zeros(Sys_dof,Sys_dof);   % initialization of system stiffness matrix
mm=zeros(Sys dof,Sys_dof);   % initialization of system mass matrix
ff=zeros(Sys_dof,1);         % initialization of system force vector

bcdof=zeros(Sys_dof,1);      % initializing the vector bcdof
bcval=zeros(Sys_dof,1);      % initializing the vector bcval

index=zeros(No_nel*No_dof,1);        % initialization of index vector
%------- ----------------------------------------------------------------
%  (2) calculation of constraints
%---- ----------------------------  ---- --------------------------------
[n1,n2]=size(ConNode);
                                     % calculate the constrained dofs
for ni=1:n1
  ki=ConNode(ni,1);
  bcdof((ki-1)*No_dof+1:ki*No_dof)=ConNode(ni,2:No_dof+1);
                                     % the code for constrained dofs
  bcval((ki-1)*No_dof+1:ki*No_dof)=ConVal(ni,2:No_dof+1);
                                     % the value at constrained dofs
end
%------------------------------------------------------------------------
%  (3) applied nodal loads
%------------------------------------------------------------------------
[n1,n2]=size(P);

for ni=1:n1
  ff(No_dof*(P(ni,2)-1)+P(ni,3))=P(ni,1);
end
%----------------------- --  -- -----------------------------------------
%  (4) calculate the element matrices and assembling
%------------------------------------------------------ ----------------
for iel=1:No_el              % loop for the total number of elements
  nd(1)=nodes(iel,1);        % 1st connected node for the iel-th element
  nd(2)=nodes(iel,2);        % 2nd connected node for the iel-th element
  x(1)=gcoord(nd(1),1); y(1)=gcoord(nd(1),2); z(1)=gcoord(nd(1),3);
                                         % coordinate of 1st node
  x(2)=gcoord(nd(2),1); y(2)=gcoord(nd(2),2); z(2)=gcoord(nd(2),3);
                                                   % coordinate of 2nd node
  leng=sqrt((x(2)-x(1))^2+(y(2)-y(1))^2+(z(2)-z(1))^2);
                             % length of element 'iel'
  xi(1)=(x(2)-x(1))/leng;    % cos of the local x-axis and system x-axis
  xi(2)=(y(2)-y(1))/leng;    % cos of the local x-axis and system y-axis
  xi(3)=(z(2)-z(1))/leng;    % cos of the local x-axis and system y-axis

  if Opt beam==1
```

```
    [k,m]=FrameElement31(prop,leng,xi,al,Opt_section,Opt_mass);
                    % compute element matrix for Euler-Bernoulli beam
  elseif Opt_beam==2
    [k,m]=FrameElement32(prop,leng,xi,al,Opt_section,Opt mass);
                    % compute element matrix for Timoshenko beam
  end

  index=femEldof(nd,No_nel,No_dof);
                    % extract system dofs associated with element
  kk=femAssemble1(kk,k,index);    % assemble system stiffness matrix
  mm=femAssemble1(mm,m,index);    % assemble system mass matrix
end
%-------------------------------------------------------------------------
%  (5) apply constraints and solve the matrix equation
%-------------------------------------------------------------------------
[kk0,mm0,ff0,bcdof0,bcval0,sdof0]=kkCheck1(kk,mm,ff,bcdof,bcval);
          % check the zero main elements in kk and eliminate the rows
          % and columns in equation associated with the zero main elements
[kk1,mm1,ff1,sdof1]=femApplybc1(kk0,mm0,ff0,bcdof0,bcval0);
                 % apply boundary conditions to the system equation

displmt=kk1\ff1;   % solve the matrix equation to find nodal displacements

No_dof1=No_dof/(Sys_dof/sdof1);
for ii=1:No_node
  for ij=1:No_dof1
    displmtnode(ii,ij)=displmt((ii-1)*No_dof1+ij,1);
  end
end
%-------------------------------------------------------------------------
%  (6) graphics of nodal connectivity and static deformation
%-------------------------------------------------------------------------
%-----------------------------------
%  (6.1) display the nodal connectivity
%-----------------------------------
if Opt_graphics1==1
  for iel=1:No_el            % loop for the total number of elements
    nd(1)=nodes(iel,1);      % 1st connected node for the iel-th element
    nd(2)=nodes(iel,2);      % 2nd connected node for the iel-th element
    x(1)=gcoord(nd(1),1); y(1)=gcoord(nd(1),2); z(1)=gcoord(nd(1),3);
                                           % coordinate of 1st node
    x(2)=gcoord(nd(2),1); y(2)=gcoord(nd(2),2); z(2)=gcoord(nd(2),3);
                                           % coordinate of 2nd node
    figure(1)
      plot3(x,y,z), xlabel('x'), ylabel('y'), zlabel('z'), hold on;
  end
 if Opt_graphics2~=1
```

```
    title('nodal connectivity of elements'); hold off;
  end
end
%-------------------------------
% (6.2) display the static deformation
%--------------------------------
if Opt_graphics2==1
  gcoordA=gcoord+10*displmtnode(:,1:3);
  for iel=1:No_el              % loop for the total number of elements
    nd(1)=nodes(iel,1);        % 1st connected node for the iel-th element
    nd(2)=nodes(iel,2);        % 2nd connected node for the iel-th element
    x(1)=gcoordA(nd(1),1); y(1)=gcoordA(nd(1),2); z(1)=gcoordA(nd(1),3);
                                                   % coordinate of 1st node
    x(2)=gcoordA(nd(2),1); y(2)=gcoordA(nd(2),2); z(2)=gcoordA(nd(2),3);
                                                  % coordinate of 2nd node
    figure(1)
      plot3(x,y,z,'r'), title('Deflection of the structure'),  hold on;
  end
  hold off
end
%--------------------------------------------------------------------------
% (7) print fem solutions
%--------------------------------------------------------------------------
disp('The calculation is use of:')

if Opt_beam==1
  disp('Euler-Bernoulli beam element')
elseif Opt_beam==2
  disp('Timoshenko beam element')
end
disp(' ')
disp('                  numrical solution')
disp('    node        x         y        z         θx        θy        θz ')
num=1:1:No_node;
displmts=[num' displmtnode]                   % print nodal displacements
%--------------------------------------------------------------------------
%    The end
% -------------------------------------------------------------------------


% -------------------------------------------------------------------------
% Input data for static analysis of frame structure (5.4)
% -------------------------------------------------------------------------
% (0.1) input basic data
% -------------------------------------------------------------------------
Prob title='static analysis of a 3-D frame structure';
```

```
No_el=12;                       % number of elements
No_nel=2;                       % number of nodes per element
No_dof=6;                       % number of dofs per node
No_node=8;                      % total number of nodes in system
Sys_dof=No_node*No_dof;         % total system dofs
%----------------------------------------------------------------------
%  (0.2) material and geometric properties
%----------------------------------------------------------------------
prop(1)=2.1e11;                                          % elastic modulus
prop(2)=0.3;                                             % Poisson's ratio
prop(3)=7860;             % mass density (mass per init volume)
prop(4)=0.01;             % height of beam cross-section in y direction
prop(5)=0.01;             % width of beam cross-section in z direction
prop(6)=0.01*0.01;        % cross-sectional area of the beam
prop(7)=0.01*0.01^3/12;   % moment of inertia of cross-section about axis y
prop(8)=0.01*0.01^3/12;   % moment of inertia of cross-section about axis z

prop(9)=1;                % polar moment of inertia
                          % = 0 not include the torsion deformation
                          % = 1 polar moment of inertia calculated by
                          %   h*b^3 or pi*(D^4-d^4)/32
                          % = a value for the torsion deformation
prop(10)=1;                                          % shear modulus
                          % = 0 not include the shear deformation
                          % = 1 shear modulus calculated by E/(2*(1+u))
                          % = the value for the shear deformation
Opt_section=1;            % option for correction factor for shear energy
                          % = 1 rectangular cross-section
                          % = 2 circular cross-section
%----------------------------------------------------------------------
%  (0.3) nodal coordinates
%----------------------------------------------------------------------
              % x, y, z coordinates of nodes in global coordinate system
gcoord=[ 1    0,  0,  0;
       2    4,  0,  0;
       3    4,  3,  0;
       0,  3,  0;
       5    0,  0,  2;
       6    4,  0,  2;
       7    4,  3,  2;
       8    0,  3,  2];
gcoord=[gcoord(:,2:4)];
al=0;                % angle between the reference coordinate system and
                     % the local coordinate system for the space element
%----------------------------------------------------------------------
%  (0.4) nodal connectivity of the elements
%----------------------------------------------------------------------
nodes=[ 1    1,  5;
      2    2,  6;
```

```
        3    3,  7;
        4    4,  8;
        5    5,  6;
        6    6,  7;
        7    7,  8;
        8    8,  5;
        9    1,  6;
       10    2,  7;
       11    3,  8;
       12    4,  5];
nodes=[nodes(:,2:3)];
%-----------------------------------------------------------------------
%  (0.5) applied loads
%-----------------------------------------------------------------------
                             % a load applied at node 7 in -y direction
P=[-100000,7,2];

%-----------------------------------------------------------------------
%  (0.6) boundary conditions
%-----------------------------------------------------------------------
ConNode=[ 1,  1,  1,  1,  1,  1,  1;        % code for constrained nodes
          2,  1,  1,  1,  1,  1,  1;
          3,  1,  1,  1,  1,  1,  1;
          4,  1,  1,  1,  1,  1,  1];
ConVal =[ 1,  0,  0,  0,  0,  0,  0;        % values at constrained dofs
          2,  0,  0,  0,  0,  0,  0;
          3,  0,  0,  0,  0,  0,  0;
          4,  0,  0,  0,  0,  0,  0];
% ----------------------------------------------------------------------
%    The end
% ----------------------------------------------------------------------
```

用采用 Euler Bernoulli 梁的刚架单元计算刚架的变形，结果见表 5.4.

表 5.4　刚架变形计算结果

节点	Euler Bernoulli 梁					
	u	v	w	θ_x	θ_y	θ_z
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	-4.0028×10^{-7}	-4.0409×10^{-7}	-9.8042×10^{-8}	3.1564×10^{-4}	2.3072×10^{-4}	-2.1346×10^{-3}
6	-2.8310×10^{-7}	-2.9029×10^{-2}	-1.3856×10^{-7}	1.1432×10^{-2}	-7.2355×10^{-5}	-3.7462×10^{-3}
7	4.0502×10^{-7}	-2.9030×10^{-2}	6.3487×10^{-3}	1.0884×10^{-2}	8.3766×10^{-4}	-2.1506×10^{-3}
8	2.9552×10^{-7}	-5.0444×10^{-7}	6.6800×10^{-8}	6.1078×10^{-4}	-4.6490×10^{-4}	-2.5184×10^{-3}

5.4.2 特征值问题与模态分析

【例 5.5】 对例 5.1 中的简支梁，分别用 Euler Bernoulli 梁单元和 Timoshenko 梁单元进行模态分析.

将如图 5.6 所示简支梁划分成 40 个单元进行有限元分析，用 MATLAB 编程计算，其中主要用到的 MATLAB 的函数(见 5.5 节)有：

- BeamElement1(prop,leng,Opt_mass)——用 Euler Bernoulli 梁单元计算单元的刚度矩阵 k 和质量矩阵 m.
- BeamElement2(prop,leng,Opt_mass)——用 Timoshenko 梁单元计算单元的刚度矩阵 k 和质量矩阵 m.
- femEldof(nd,No_nel,No_dof)——计算单元自由度对应到结构系统上的自由度.
- femAssemble1(kk,k,index)——用于将单元刚度矩阵或质量矩阵组集到系统的总体刚度矩阵 kk 或总体质量矩阵 mm.
- femApplybc1(kk,mm,ff,bcdof,bcval)——对结构系统施加边界条件.
- kkCheck1(kk,mm,ff,bcdof,bcval)——检查总体刚度矩阵 kk 的主元是否为零.
- bcCheck1(kk0,mm0,ff0,bcdof0,bcval0)——检查边界条件，消去相应的自由度.

M 文件如下：

```
%--------------------------------------------------------------------------
% Example 5.5
%   To solve the eigenvalue problem for a beam structure.
%   using Euler Bernoulli beam or Timoshenko beam elements.
%   nodal dofs: {v1  θ  v2  θ2}
% Problem description
%   Find the natural frequencies and modal shapes of a simply supported
%   beam whose length is 1 m. The beam has elastic modulus of 2.1x10^11
%   and its cross-section is 0.02 m height by 0.02 m width. The mass
%   density is 7860 kg/m^3. A concentrated load of -500 N is applied at
%   the middle of the beam.
% Variable descriptions
%   k, m - element stiffness matrix and element mass matrix
%   kk, mm - system stiffness matrix and system mass matrix
%   ff - system force vector
%   index - a vector containing system dofs associated with each element
%   bcdof - a vector containing dofs associated with boundary conditions
%   bcval - a vector containing boundary condition values associated
%   with the dofs in bcdof
%--------------------------------------------------------------------------
%  (0) input data
%--------------------------------------------------------------------------
clear; clc;

Beam_InputData541;   % import the input data for the information of
```

```
                        % nodes, elements, loads, constraints and materials
Opt_beam=1;                          % option for type of the beam:
                                     % =1 Euler Bernoulli beam
                                     % =2 Timoshenko beam
Opt_mass=1;                          % option for mass matrix:
                                     % =1 consistent mass matrix
                                     % =2 lumped mass matrix
Opt_graphics1=0;               % option for graphics of the nodal connectivity
Opt_graphics3=1;               % option for graphics of the modal shape

ith=3;                         % select the order of modes to display
%------------------------------------------------------------------------
%  (1) initialization of matrices and vectors to zero
%------------------------------------------------------------------------
k=zeros(No_nel*No_dof,No_nel*No_dof);       % element stiffness matrix
m=zeros(No_nel*No_dof,No_nel*No_dof);       % element mass matrix

kk=zeros(Sys_dof,Sys_dof);   % initialization of system stiffness matrix
mm=zeros(Sys_dof,Sys_dof);   % initialization of system mass matrix
ff=zeros(Sys_dof,1);         % initialization of system force vector

index=zeros(No_nel*No_dof,1);       % initialization of index vector

bcdof=zeros(Sys_dof,1);             % initializing the vector bcdof
bcval=zeros(Sys_dof,1);             % initializing the vector bcval
%------------------------------------------------------------------------
%  (2) calculation of constraints
%------------------------------------------------------------------------
[n1,n2]=size(ConNode);
                                    % calculate the constrained dofs
for ni=1:n1
  ki=ConNode(ni,1);
  bcdof((ki-1)*No_dof+1:ki*No_dof)=ConNode(ni,2:No_dof+1);
                                     % the code for constrained dofs
  bcval((ki-1)*No_dof+1:ki*No_dof)=ConVal(ni,2:No_dof+1);
                                     % the value at constrained dofs
end
%------------------------------------------------------------------------
%  (3) applied nodal loads
%------------------------------------------------------------------------

ff(No_dof*(P(2)-1)+P(3))=P(1);

%------------------------------------------------------------------------
%  (4) calculate the element matrices and assembling
%------------------------------------------------------------------------
for iel=1:No_el                 % loop for the total number of elements
```

```
  nd(1)=iel;                    % 1st node number of the iel-th element
  nd(2)=iel+1;                  % 2nd node number of the iel-th element
  x(1)=gcoord(nd(1),1); y(1)=gcoord(nd(1),2); z(1)=gcoord(nd(1),3);
                                              % coordinates of 1st node
  x(2)=gcoord(nd(2),1); y(2)=gcoord(nd(2),2); z(2)=gcoord(nd(2),3);
                                              % coordinates of 2nd node
  leng=sqrt((x(2)-x(1))^2+(y(2)-y(1))^2+(z(2)-z(1))^2);
                                              % length of element 'iel'
  if Opt_beam==1
    [k,m]=BeamElement11(prop,leng,Opt_mass);
                    % compute element matrices for Euler-Bernoulli beam
  else
    [k,m]=BeamElement12(prop,leng,Opt_mass);
                    % compute element matrices for Timoshenko beam
  end

  index=femEldof(nd,No_nel,No_dof); % extract system dofs associated with element

  kk=femAssemble1(kk,k,index);        % assemble system stiffness matrix
  mm=femAssemble1(mm,m,index);        % assemble system mass matrix
end
%-----------------------------------------------------------------------
%  (5) apply constraints and solve the matrix equation
%-----------------------------------------------------------------------
[kk0,mm0,ff0,bcdof0,bcval0,sdof0]=kkCheck1(kk,mm,ff,bcdof,bcval);
                                   % check the zero main elements in kk
[kk1,mm1,ff1,sdof1]=femApplybc1(kk0,mm0,ff0,bcdof0,bcval0);
                        % apply boundary conditions to the system equation
[kk2,mm2,ff2,sdof2]=bcCheck1(kk0,mm0,ff0,bcdof0,bcval0);
      % check the boundary conditions in equation and eliminate the rows
      % and columns associated with the boundary conditions
[V,D]=eig(kk2,mm2);             % solve the eigenvalue problem of matrix
[lambda,ki]=sort(diag(D));      % sort the eigenvaules and eigenvectors
omega=sqrt(lambda);             % the frequency vector in radon/sec.
omega1=sqrt(lambda)/(2*pi);          % the frequency vector in Hz.
V=V(:,ki);                                        % the modal matrix
%-----------------------------------------------------------------------
%  (6) Analytical solution
%-----------------------------------------------------------------------
E prop(1); Iz=prop(8); rho=prop(3)*prop(6); L=1;
i=(1:sdof2)';
omega2=i.*i*pi^2*sqrt(E*Iz/(rho*L^4));
omega3=omega2/(2*pi);
%-----------------------------------------------------------------------
%  (7) graphics of nodal connectivity and static deformation
```

```
%-------------------------------------------------------------------------
%----------------------------------
% (7.1) display the nodal connectivity
%----------------------------------
if Opt_graphics1==1
  for iel=1:No_el                % loop for the total number of elements
    nd(1)=iel;                   % 1st node number of the iel-th element
    nd(2)=iel+1;                 % 2nd node number of the iel-th element
    x(1)=gcoord(nd(1),1); y(1)=gcoord(nd(1),2); z(1)=gcoord(nd(1),3);
                                                  % coordinates of 1st node
    x(2)=gcoord(nd(2),1); y(2)=gcoord(nd(2),2); z(2)=gcoord(nd(2),3);
                                                  % coordinates of 2nd node
    figure(1)
      plot(x,y); xlabel('x'), ylabel('y'), hold on;
      axis([-0.1,1.1,-1.5,1.5]);
      title('nodal connectivity of elements');
  end
  hold off
end
%-------------------------------------------------------------------------
% (7.3) draw the modal shape
%-------------------------------------------------------------------------
if Opt_graphics3==1
  jk=ith; Vi=[V(:,jk)];           % chose the order of modes to display
  ik=0;          % initial the counter for locating the modal coordinates

  for ii=1:sdof0                  % loop for the total number of sdof
    if bcdof0(ii)==0
      mcoord(ii,1)=Vi(ii-ik);
    else
      mcoord(ii,1)=0;
      ik=ik+1;
    end
  end

  for ii=1:No_node                % loop for the total number of nodes
    for ij=1:No dof1
      mcoordA(ii,ij)=mcoord((ii-1)*No_dof1+ij,1);
    end
  end

  nv=20;

  for i=1:nv+1
    t=(i-1)*(2*pi)/20;
    mcoordB=gcoord+[zeros(No_node,1),mcoordA(:,1),zeros(No node,1)]*cos(t);
```

```
    for iel=1:No_el                % loop for the total number of elements
      nd(1)=iel;                   % starting node number for element 'iel'
      nd(2)=iel+1;                 % ending node number for element 'iel'
      x(1)=mcoordB(nd(1),1); y(1)=mcoordB(nd(1),2); z(1)=mcoordB(nd(1),3);
                                                      % coordinate of 1st node
      x(2)=mcoordB(nd(2),1); y(2)=mcoordB(nd(2),2); z(2)=mcoordB(nd(2),3);
                                                      % coordinate of 2nd node
      figure(2)
        plot(x,y,'b'), xlabel('x'), ylabel('y'), hold on;
        axis([-0.1,1.1,-1.5,1.5])
        title([num2str(jk), 'th modal shape of the structure']);
    end
    hold off
    M(:,i)=getframe;
  end
movie(M,5,10);
end
%--------------------------------------------------------------------------
%  (8) print fem solutions
%--------------------------------------------------------------------------
disp('The calculation is use of:')

if Opt_beam==1
  disp('Euler-Bernoulli beam element')
else
  disp('Timoshenko beam element')
end

if Opt_mass==1
  disp('and consistent mass matrix')
elseif Opt_mass==2
  disp('and lumped mass matrix')
end

disp(' ')
disp('    mode   numrical   analytical')

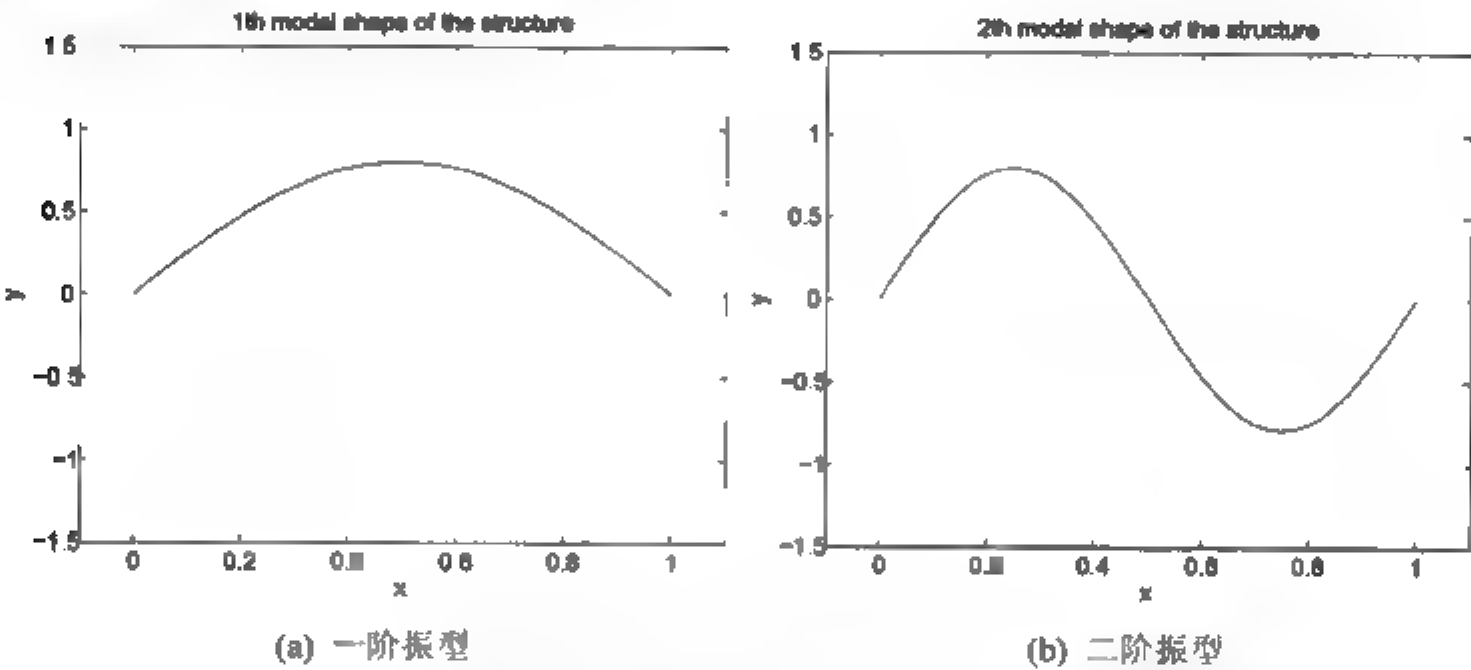
num=1:1:10;                                 % print natural frequencies
frequency=[num' omega1(1:10)  omega3(1:10)]
%--------------------------------------------------------------------------
%     The end
%--------------------------------------------------------------------------
```

分别用 Euler Bernoulli 梁单元和 Timoshenko 梁单元进行模态分析，简支梁前 10 阶固有频率见表 5.5.

表 5.5 简支梁前 10 阶固有频率

阶 数	有限元解		解 析 解
	Euler Bernoulli 梁	Timoshenko 梁	
1	46.8768	46.8888	46 8768
2	187.5071	187.7003	187.5071
3	421.8918	422.8679	421.8909
4	750.0333	753.1108	750.0282
5	1171.9387	1179.4286	1171.9191
6	1687.6209	1703.0923	1687.5635
7	2297.1059	2325.6338	2296.9614
8	3000.4340	3048.8297	3000.1129
9	3797.6669	3874.6813	3797.0179
10	4688.8936	4805.3873	4687 6764

前一阶、二阶振型如图 5.9 所示.



(a) 一阶振型 (b) 二阶振型

图 5.9 简支梁的低阶振型

【例 5.6】 对于例 5.4 中的空间刚架结构，用模态分析方法分别计算结构在脉冲、阶跃和简谐激励下的响应. 脉冲、阶跃和简谐激励函数分别为：$F(t)=F_0\delta(t)$，$F(t)=F_0$，$F(t)-F_0\cos\omega t$，其中，F_0是在结构上节点 7 沿 y 方向作用 100 000 N 的集中载荷形成的力向量，$\omega=300$ rad/s.

与例 5.4 相同将刚架划分成 12 个单元进行有限元分析，用 MATLAB 编程计算，其中主要用到的 MATLAB 函数(见 5.5 节)有：

- FrameElement31(prop,leng,beta,xi,al,Opt_section,Opt_mass)——用 Euler Bernoulli 梁计算单元的刚度矩阵 k 和质量矩阵 m.
- FrameElement32(prop,leng,beta,xi,al,Opt_section,Opt_mass) ——用 Timoshenko 梁

计算单元的刚度矩阵 k 和质量矩阵 m.

- femEldof(nd,No_nel,No_dof)——计算单元自由度对应到结构系统上的自由度.
- femAssemble1(kk,k,index)——用于将单元刚度矩阵或质量矩阵组集到系统的总体刚度矩阵 kk 或总体质量矩阵 mm.
- femApplybc1(kk,mm,ff,bcdof,bcval)——对结构系统施加边界条件.
- kkCheck1(kk,mm,ff,bcdof,bcval)——检查总体刚度矩阵 kk 的主元是否为零.
- bcCheck1(kk0,mm0,ff0,bcdof0,bcval0)——检查边界条件，消去相应的自由度.
- ImpulseRespt(kk,mm,fd,u,t,C,q0,dq0,a,b)——计算结构在脉冲激励下的响应.
- StepRespt(kk,mm,fd,u,t,C,q0,dq0,a,b)——计算结构在阶跃激励下的响应.
- HarmonicRespt(kk,mm,fd,omega0,t,C,q0,dq0,a,b)——计算结构在简谐激励下的响应.

M 文件如下：

```
%-----------------------------------------------------------------------
% Example 5.6
%   To solve the dynamic response of a space frame structure.
%   nodal dof: {u1 v1 w1 θx1 θy1 θz1 u2 v2 w2 θx2 θy2 θz2}
% Problem description
%   Find the dynamic response of a space frame structure as shown in
%   Fig.5-10, which is made of twelve beam members. The frame is of the
%   length of 4 m, the width of 3 m and the height of 2 m. All members
%   have cross-section of 0.01 m height by 0.01 m width.
%   The elastic modulus of the structure material is 2.1e11 Pa. The mass
%   density is 7860 kg/m3. The frame is subjected to a concentrated load
%   of 10000N at node 7 in -y direction.
% Variable descriptions
%   k,m = element stiffness matrix and element mass matrix
%   kk,mm = system stiffness matrix and system mass matrix
%   ff = system force vector
%   index = a vector containing system dofs associated with each element
%   bcdof = a vector containing dofs associated with boundary conditions
%   bcval = a vector containing boundary condition values associated
%   with the dofs in 'bcdof'
%-----------------------------------------------------------------------
%  (0) input control data
%-----------------------------------------------------------------------
clear; clc;

Beam_InputData544;   % import the input data for the information of
                     % nodes, elements, loads, constraints and materials
Opt_beam=1;                              % option for type of the beam
                                          % =1 Euler Bernoulli beam
                                          % =2 Timoshenko beam
Opt_mass=1;                               % option for mass matrix
                                          % =1 consistent mass matrix
                                          % =2 lumped mass matrix
```

```
Opt section=1;                        % option for type of cross-section
                                      % = 1 rectangular cross-section
                                      % = 2 circular cross-section
TypeResp=1;             % option for selecting the type of the response
                        % =1 impulse response:  ImpulseRespt.m
                        % =2 step response:     StepRespt.m
                        % =3 harmonic response: HarmonicRespt.m

dt=0.0005;                            % time step size
ti=0;                                 % initial time
tf=0.25;                              % final time
nt=fix((tf-ti)/dt);                   % number of time steps
tt=ti:dt:ti+nt*dt;                    % generate time vector

ac=0.0002; bc=0.00008;          % Parameters for proportional damping

al=0;             % angle between the reference coordinate system and
                  % the local coordinate system for the space element
omega0=100;                     % frequency of the applied force
%-----------------------------------------------------------------------
%  (1) initialization of matrices and vectors to zero
%-----------------------------------------------------------------------
k=zeros(No_nel*No_dof,No_nel*No_dof);       % element stiffness matrix
m=zeros(No_nel*No_dof,No_nel*No_dof);        % element mass matrix

kk=zeros(Sys_dof,Sys_dof);   % initialization of system stiffness matrix
mm=zeros(Sys_dof,Sys_dof);   % initialization of system mass matrix
ff=zeros(Sys_dof,1);         % initialization of system force vector

bcdof=zeros(Sys_dof,1);             % initializing the vector bcdof
bcval=zeros(Sys_dof,1);             % initializing the vector bcval

index=zeros(No_nel*No_dof,1);       % initialization of index vector
%-----------------------------------------------------------------------
%  (2) calculation of constraints
%-----------------------------------------------------------------------
[n1,n2]=size(ConNode);
                              % calculate the constrained dofs
for ni=1:n1
  ki=ConNode(ni,1);
  bcdof((ki-1)*No_dof+1:ki*No_dof)=ConNode(ni,2:No_dof+1);
                              % the code for constrained dofs
  bcval((ki-1)*No_dof+1:ki*No_dof)=ConVal(ni,2:No_dof+1);
                              % the value at constrained dofs
end
%-----------------------------------------------------------------------
%  (3) applied nodal loads
```

```
%-------------------------------------------------------------------
[n1,n2]=size(P);

for ni=1:n1
  ff(No_dof*(P(ni,2)-1)+P(ni,3))=P(ni,1);
end
%-------------------------------------------------------------------
% (4) calculate the element matrices and assembling
%-------------------------------------------------------------------
for iel=1:No_el               % loop for the total number of elements
  nd(1)=nodes(iel,1);         % 1st connected node for the iel-th element
  nd(2)=nodes(iel,2);        % 2nd connected node for the iel-th element
  x(1)=gcoord(nd(1),1); y(1)=gcoord(nd(1),2); z(1)=gcoord(nd(1),3);
                                        % coordinate of 1st node
  x(2)=gcoord(nd(2),1); y(2)=gcoord(nd(2),2); z(2)=gcoord(nd(2),3);
                                       % coordinate of 2nd node
  leng=sqrt((x(2)-x(1))^2+(y(2)-y(1))^2+(z(2)-z(1))^2);
                              % length of element 'iel'
  xi(1)=(x(2)-x(1))/leng;     % cos of the local x-axis and system x-axis
  xi(2)=(y(2)-y(1))/leng;     % cos of the local x-axis and system y-axis
  xi(3)=(z(2)-z(1))/leng;     % cos of the local x-axis and system y-axis

  if Opt_beam==1
    [k,m]=FrameElement31(prop,leng,xi,al,Opt_section,Opt_mass);
                        % compute element matrix for Euler-Bernoulli beam
  elseif Opt_beam==2
    [k,m]=FrameElement32(prop,leng,xi,al,Opt_section,Opt_mass);
                        % compute element matrix for Timoshenko beam
  end

  index=femEldof(nd,No_nel,No_dof);
  % extract system dofs associated with element

  kk=femAssemble1(kk,k,index);      % assemble system stiffness matrix
  mm=femAssemble1(mm,m,index);      % assemble system mass matrix
end
%-------------------------------------------------------------------
% (5) apply constraints and solve the matrix equation
%-------------------------------------------------------------------
 [kk0,mm0,ff0,bcdof0,bcval0,sdof0]=kkCheck1(kk,mm,ff,bcdof,bcval);
                                   % check the zero main elements in kk
[kk1,mm1,ff1,sdof1]=femApplybc1(kk0,mm0,ff0,bcdof0,bcval0);
           % apply boundary conditions to the system equation
 [kk2,mm2,ff2,sdof2]=bcCheck1(kk0,mm0,ff0,bcdof0,bcval0);
           % check the boundary conditions in equation and eliminate the
           % rows and columns associated with the boundary conditions
%-------------------------------------------------------------------
```

```
[V,D]=eig(kk2,mm2);            % solve the eigenvalue problem of matrix
[lambda,ki]=sort(diag(D));     % sort the eigenvaules and eigenvectors
omega=sqrt(lambda);            % the frequency vector in rad/s
omega1=sqrt(lambda)/(2*pi);              % the frequency vector in Hz

V=V(:,ki);                                             % the modal matrix
%------------------------------------------------------------------------
% (6) calculate dynamic response
%------------------------------------------------------------------------
                       % present the initial computation conditions
u=1;
C=eye(sdof2);
q0=zeros(sdof2,1);                            % initial displacement
dq0=zeros(sdof2,1);                             % initial velocity

if TypeResp==1
  [eta,y,omega2,sdof3]=ImpulseRespt(kk2,mm2,ff2,u,tt,C,q0,dq0,ac,bc);
                                           % compute the impulse responses
elseif TypeResp==2
  [eta,y,omega2,sdof3]=StepRespt(kk2,mm2,ff2,u,tt,C,q0,dq0,ac,bc);
                                             % compute the step responses
else
  [eta,y,omega2,sdof3]=HarmonicRespt(kk2,mm2,ff2,omega0,tt,C,q0,dq0,ac,bc);
                                          % compute the harmonic responses
end
%------------------------------------------------------------------------
%  (7) graphics of dynamic response
%------------------------------------------------------------------------
jth=14;  tpoint=2;
   % Plot the graph of the time-history response at jth degree of freedom
plot(tt,y(jth,:))
xlabel('Time (seconds)')
ylabel('Nodal  displmt. (m)')
%------------------------------------------------------------------------
%  (8) print fem solutions
%------------------------------------------------------------------------
disp('The calculation is use of:')

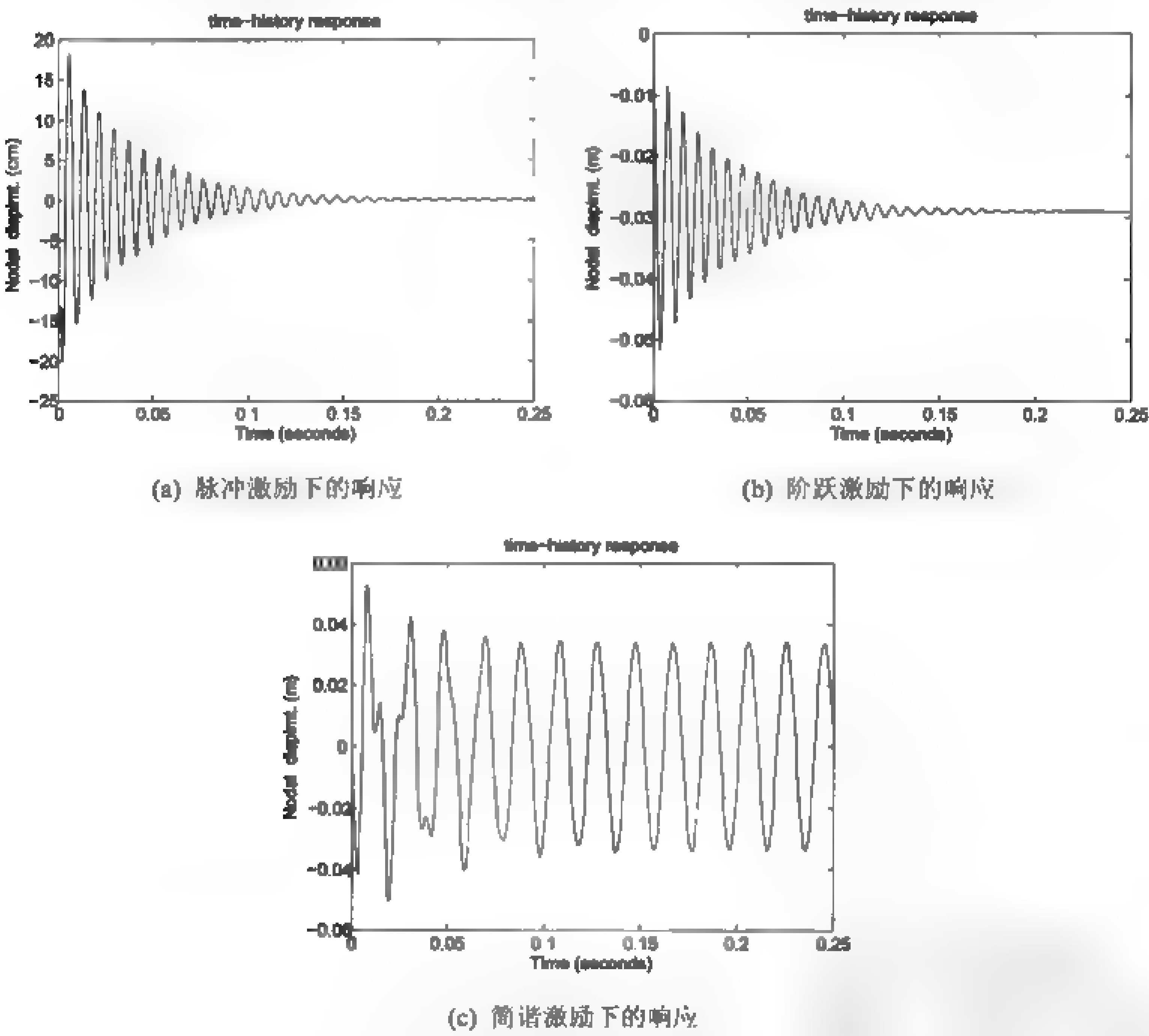
if Opt_beam==1
  disp('Euler-Bernoulli beam element')
elseif Opt_beam==2
  disp('Timoshenko beam element')
end

if Opt_mass==1
  disp('and consistent mass matrix')
elseif Opt_mass==2
```

```
  disp('and lumped mass matrix')
end

num=1:1:sdof2;
frequency=[num' omega1]                    % print natural frequency
% ---------------------------------------------------------------------
%    The end
% ---------------------------------------------------------------------
```

用 Euler Bernoulli 梁单元进行计算，结构在节点 7 y 方向的响应如图 5.10 所示.



(a) 脉冲激励下的响应

(b) 阶跃激励下的响应

(c) 简谐激励下的响应

图 5.10　振型叠加法计算的刚架结构节点 7 处响应

5.4.3　瞬态问题分析

【例 5.7】 对于例 5.3 中的平面刚架结构，梁的横截面面积为 $0.10\times0.05\text{m}^2$，梁的弹性模量为 2.1×10^{11}Pa，质量密度为 7 860kg/m^3. 用直接积分方法计算结构在脉冲激励下的瞬态响应. 脉冲激励函数为：$F(t)=F_0\delta(t)$，其中，F_0 是在结构中部沿$-y$ 方向作用力的幅

值，$F_0=-500\text{N}$.

与例 5.3 相同将刚架划分成 11 个单元进行有限元分析，激励载荷作用在节点 6 上，用 MATLAB 编程计算，其中主要用到的 MATLAB 函数(见 5.5 节)有：

- FrameElement31(prop,leng,beta,xi,al,Opt_section,Opt_mass) ——用 Euler Bernoulli 梁计算单元的刚度矩阵 k 和质量矩阵 m.
- FrameElement32(prop,leng,beta,xi,al,Opt_section,Opt_mass) ——用 Timoshenko 梁计算单元的刚度矩阵 k 和质量矩阵 m.
- femEldof(nd,No_nel,No_dof)——计算单元自由度对应到结构系统上的自由度.
- femAssemble1(kk,k,index)——用于将单元刚度矩阵或质量矩阵组集到系统的总体刚度矩阵 kk 或总体质量矩阵 mm.
- femApplybc1(kk,mm,ff,bcdof,bcval)——对结构系统施加边界条件.
- kkCheck1(kk,mm,ff,bcdof,bcval)——检查总体刚度矩阵 kk 的主元是否为零.
- mmCheck1(kk0,mm0,ff0,bcdof0,bcval0)——检查总体质量矩阵 mm 的主元是否为零.
- bcCheck1(kk0,mm0,ff0,bcdof0,bcval0)——检查边界条件，消去相应的自由度.
- TransResp1(kk1,cc1,mm1,ft0,bcdof1,nt,dt,q0,dq0)——中心差分法计算响应.
- TransResp3(kk1,cc1,mm1,ft0,bcdof1,nt,dt,q0,dq0)——Houbolt 法计算响应.
- TransResp4(kk1,cc1,mm1,ft0,bcdof1,nt,dt,q0,dq0)——Wilson-θ 法计算响应.
- TransResp5(kk1,cc1,mm1,ft0,bcdof1,nt,dt,q0,dq0)——Newmark 法计算响应.

M 文件如下：

```
%----------------------------------------------------------------------%
% Example 5.7
%   To solve transient response of a 2-d frame structure.
%   The solution methods are: 1) central difference scheme. 3) Houbolt
%   integration scheme.
%   4) Wilson ¦È integration scheme. 5) Newmark integration scheme
%   nodal dof: {u1  v1  w1 θx1 θy1 θz1  u2  v2  w2 θx2 θy2 θz2}
% Problem description
%   Find the response of a frame structure which is made of three beams
%   of lengths of 4 m, 3 m and 4 m, respectively. All beams have cross-
%   section of 0.10 m height by 0.05 m width. The elastic modulus is
%   2.10x10^11 Pa. The frame is subjected to an impulse load of amplitude
%   500 N in the middle of the top beam. One end of the each vertical
%   beam is fixed. (see Fig. 5-9 for the element discretization)
% Variable descriptions
%   k, m - element stiffness matrix and mass matrix
%   kk, mm - system stiffness matrix and mass matrix
%   ff - system force vector
%   index - a vector containing system dofs associated with each element
%   bcdof - a vector containing dofs associated with boundary conditions
%   bcval - a vector containing boundary condition values associated
%   with the dofs in 'bcdof'
```

```
%   dsp - displacement matrix
%   vel - velocity matrix
%   acc - acceleartion matrix
%-----------------------------------------------------------------------
%  (0) input control data
%-----------------------------------------------------------------------
clear; clc;

Beam_InputData547;       % import the input data for the information of
                     % nodes, elements, loads, constraints and materials
Opt_beam=1;                         % option for type of the beam
                                    % =1 Euler Bernoulli beam
                                    % =2 Timoshenko beam
Opt_mass=2;                         % option for mass matrix
                                    % =1 consistent mass matrix
                                    % =2 lumped mass matrix
Opt_section=1;                      % option for type of cross-section
                                    % = 1 rectangular cross-section
                                    % = 2 circular cross-section
TypeMethod=1;               % option for selecting the solution method
                            % = 1 central difference scheme
                            % = 3 Houbolt integration scheme
                            % = 4 Wilson θ integration scheme
                            % = 5 Newmark integration scheme
Typeload=1;                 % option for selecting the load type
                                                % = 1 impulse load
                                                  % = 2 step load
                                               % = 3 Harmonic load

dt=0.00001;                                          % time step size
ti=0;                                                % initial time
tf=0.200;                                            % final time
nt=fix((tf-ti)/dt);                         % number of time steps
tt=ti:dt:ti+nt*dt;                     % generate time samples vector

ac=0.00002; bc=0.00008;         % Parameters for proportional damping

al=0;                 % angle between the reference coordinate system and
                    % the local coordinate system for the space element
%-----------------------------------------------------------------------
%  (1) initialization of matrices and vectors to zero
%-----------------------------------------------------------------------
k=zeros(No_nel*No_dof,No_nel*No_dof);        % element stiffness matrix
m=zeros(No_nel*No_dof,No_nel*No_dof);        % element mass matrix

kk=zeros(Sys_dof,Sys_dof);  % initialization of system stiffness matrix
mm=zeros(Sys_dof,Sys_dof);   % initialization of system mass matrix
```

```
ff=zeros(Sys_dof,1);          % initialization of system force vector

bcdof=zeros(Sys_dof,1);            % initializing the vector bcdof
bcval=zeros(Sys_dof,1);             % initializing the vector bcval

index=zeros(No_nel*No_dof,1);       % initialization of index vector
%-------------------------------------------------------------------------
%  (2) calculation of constraints
%-------------------------------------------------------------------------
[n1,n2]=size(ConNode);
                                   % calculate the constrained dofs
for ni=1:n1
  ki=ConNode(ni,1);
  bcdof((ki-1)*No_dof+1:ki*No_dof)=ConNode(ni,2:No_dof+1);
                                  % the code for constrained dofs
  bcval((ki-1)*No_dof+1:ki*No_dof)=ConVal(ni,2:No_dof+1);
                                  % the value at constrained dofs
end
%-------------------------------------------------------------------------
%  (3) applied nodal loads
%-------------------------------------------------------------------------
[n1,n2]=size(P);

for ni=1:n1
  ff(No_dof*(P(ni,2)-1)+P(ni,3))=P(ni,1);
end
%-------------------------------------------------------------------------
%  (4) calculate the element matrices and assembling
%-------------------------------------------------------------------------
for iel=1:No_el                % loop for the total number of elements
  nd(1)=iel;                 % starting node number for element 'iel'
  nd(2)=iel+1;                % ending node number for element 'iel'
  x(1)=gcoord(nd(1),1); y(1)=gcoord(nd(1),2); z(1)=gcoord(nd(1),3);
                                              % coordinate of 1st node
  x(2)=gcoord(nd(2),1); y(2)=gcoord(nd(2),2); z(2)=gcoord(nd(2),3);
                                             % coordinate of 2nd node
  leng=sqrt((x(2)-x(1))^2+(y(2)-y(1))^2+(z(2)-z(1))^2);
  xi(1)=(x(2)-x(1))/leng;     % cos of the local x-axis and system x-axis
  xi(2)=(y(2)-y(1))/leng;     % cos of the local x-axis and system y-axis
  xi(3)=(z(2)-z(1))/leng;     % cos of the local x-axis and system y-axis

  if Opt_beam==1
    [k,m]=FrameElement31(prop,leng,xi,al,Opt_section,Opt_mass);
                       % compute element matrix for Euler-Bernoulli beam
  elseif Opt_beam==2
    [k,m]=FrameElement32(prop,leng,xi,al,Opt_section,Opt mass);
                      % compute element matrix for Timoshenko beam
```

```
  end

  index=femEldof(nd,No_nel,No dof);
  % extract system dofs associated with element
  kk=femAssemble1(kk,k,index);      % assemble system stiffness matrix
  mm=femAssemble1(mm,m,index);      % assemble system mass matrix

end
%-------------------------------------------------------------------------
%  (5) apply constraints and solve the matrix equation
%-------------------------------------------------------------------------
[kk0,mm0,ff0,bcdof0,bcval0,sdof0]=kkCheck1(kk,mm,ff,bcdof,bcval);
       % check the zero main elements in kk and eliminate the rows
       % and columns in equation associated with the zero main elements
[kk1,mm1,ff1,sdof1]=femApplybc1(kk0,mm0,ff0,bcdof0,bcval0);
         % apply boundary conditions to the system equation

[kk2,mm2,ff2,sdof2]=bcCheck1(kk0,mm0,ff0,bcdof0,bcval0);
         % check the boundary conditions in equation and eliminate
         % the rows and columns associated with the boundary conditions

[V,D]=eig(kk2,mm2);               % solve the eigenvalue problem of matrix
[lambda,ki]=sort(diag(D));         % sort the eigenvaules and eigenvectors
omega=sqrt(lambda);                  % the frequency vector in radin/s
omega1=sqrt(lambda)/(2*pi);             % the frequency vector in Hz

V=V(:,ki);                                            % the modal matrix
%-------------------------------------------------------------------------
%  (6) Check the eigenvalues and the damping parameters
%-------------------------------------------------------------------------
             % check whether the eigenvalues are infinite and eliminate
             % the eigenvectors associated with the bad eigenvalues.
jk=0;

for ii=1:sdof2                  % loop for find the infinite in omega
  check=omega(ii);
  if check>1.0e12
    jk=jk+1;                   % location of the infinite frequency
    omi(jk)=ii;          % storing the location of the infinite frequency
  end
end

sdof3=sdof2-jk;
V1=[V(:,1:sdof3)];                        % truncate the modal vectors

Factor=diag(V1'*mm2*V1);
Vnorm=V1*inv(sqrt(diag(Factor)));     %  Eigenvectors are normalized
```

```
%--------------------------------------------------------------------------
omega2=diag(sqrt(Vnorm'*kk2*Vnorm));          % Natural frequencies

Modamp=Vnorm'*(ac*mm2+bc*kk2)*Vnorm;   % Form the Rayleigh damping
zeta=diag((1/2)*Modamp*inv(diag(omega2)));      % The damping ratio

if (max(zeta) >= 1)
  disp('Warning - Your maximu damping ratio is grater than or equal to 1')
  disp('You have to reselect ac and bc ')
  pause
  disp('If you want to continue, type return key')
end
%--------------------------------------------------------------------------
%  (7) calculate transient response
%--------------------------------------------------------------------------
[kk1,mm1,ff1,bcdof1,bcval1,sdof1]=mmCheck1(kk0,mm0,ff0,bcdof0,bcval0);
           % check the zero main elements in mm and eliminate the rows
           % and columns in equation associated with the zero main
elements
switch Typeload
  case 1                              % Impulse force function
    u=[1,zeros(1,nt)];
    ft0=ff1*u;
  case 2                                % Step force function
    u(1,1:nt+1)=1;
    ft0=ff1*u;
  case 3                             % Harmonic force function
    u=cos(omega0*tt);
    ft0=ff1*u;
  otherwise
    ft0=ff1;                            % a given force function
end

cc1=ac*mm1+bc*kk1;              % Form the Rayleigh damping matrix

q0=zeros(sdof1,1); dq0=zeros(sdof1,1);  % initial displacement and velocity

switch TypeMethod
  case 1                          % central difference scheme 1
    [acc,vel,dsp]=TransResp1(kk1,cc1,mm1,ft0,bcdof1,nt,dt,q0,dq0);
  case 3                          % Houbolt integration scheme
    [acc,vel,dsp]=TransResp3(kk1,cc1,mm1,ft0,bcdof1,nt,dt,q0,dq0);
  case 4                          % Wilson ¦È integration scheme
    [acc,vel,dsp]=TransResp4(kk1,cc1,mm1,ft0,bcdof1,nt,dt,q0,dq0);
  case 5                          % Newmark integration scheme
    [acc,vel,dsp]=TransResp5(kk1,cc1,mm1,ft0,bcdof1,nt,dt,q0,dq0);
  otherwise
```

```
    disp('Unknown method.')
end
%--  ----------------        ----------------------------------------------
%  (8) graphics of dynamic response
%--------------------------------------------------------------------------
  jth=14;
    % Plot the graph of the time-history response at jth degree of freedom
  plot(tt,dsp(jth,:))
  xlabel('Time  (seconds)'), ylabel('displacement  (m)')
  title('time-history response')
%--------------------------------------------------------------------------
%  (9) print fem solutions
%--------------------------------------------------------------------------
switch Typeload
  case 1
    disp('The excitation is impulse force')
  case 2
    disp('The excitation is step force')
  case 3
    disp('The excitation is step force')
    otherwise
    disp('The given foece')
end

disp('The calculation is use of:')

if Opt_beam==1
  disp('Euler-Bernoulli beam element')
elseif Opt_beam==2
  disp('Timoshenko beam element')
end

if Opt_mass==1
  disp('and consistent mass matrix')
elseif Opt_mass==2
  disp('and lumped mass matrix')
end

num=1:1:sdof2;
frequency=[num' omega1]                    % print natural frequency
%-------------------- -----------------------------------------------------
%    The end
%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
% Input data for static analysis of frame structure (5.7)
%--------------------------------------------------------------------------
%  (0.1) input basic data
```

```
%-----------------------------------------------------------------------
Prob_title='transient analysis of a 2-D frame structure';

No_el=11;                          % number of elements
No_nel=2;                          % number of nodes per element
No_dof=6;                          % number of dofs per node
No_node=12;                        % total number of nodes in system
Sys_dof=No_node*No_dof;            % total system dofs
%-----------------------------------------------------------------------
%  (0.2) material and geometric properties
%-----------------------------------------------------------------------
prop(1)=2.1e11;                                   % elastic modulus
prop(2)=0.3;                                      % Poisson's ratio
prop(3)=7860;                     % mass density (mass per init volume)
prop(4)=0.10;          % height of beam cross-section in y direction
prop(5)=0.05;          % width of beam cross-section in z direction
prop(6)=0.05*0.10;                    % cross-sectional area of the beam
prop(7)=0;            % moment of inertia of cross-section about axis y
prop(8)=0.05*0.10^3/12;  % moment of inertia of cross-section about axis z

prop(9)=0;                              % polar moment of inertia
                    % = 0 not include the torshional deformation
                    % = 1 polar moment of inertia calculated by
                           %     h*b^3 or pi*(D^4-d^4)/32
                    % = the value for the torshional deformation
prop(10)=1;                                   % shear modulus
                      % = 0 not include the shear deformation
                  % = 1 shear modulus calculated by E/(2*(1+u))
                         % = a value for the shear deformation
Opt_section=1;          % option for correction factor for shear energy
                            % = 1 rectangular cross-section
                                       % = 2 circular cross-section
%-----------------------------------------------------------------------
%  (0.3) nodal coordinates
%-----------------------------------------------------------------------
                 % x, y, z coordinates in global coordinate system
gcoord=[ 1    0.0     0.0     0.0;
         2    0.0     1.0     0.0;
         3    0.0     2.0     0.0;
         4    0.0     3.0     0.0;
         5    0.0     4.0     0.0;
         6    1.0     4.0     0.0;
         7    2.0     4.0     0.0;
         8    3.0     4.0     0.0;
         9    3.0     3.0     0.0;
        10    3.0     2.0     0.0;
        11    3.0     1.0     0.0;
        12    3.0     0.0     0.0];

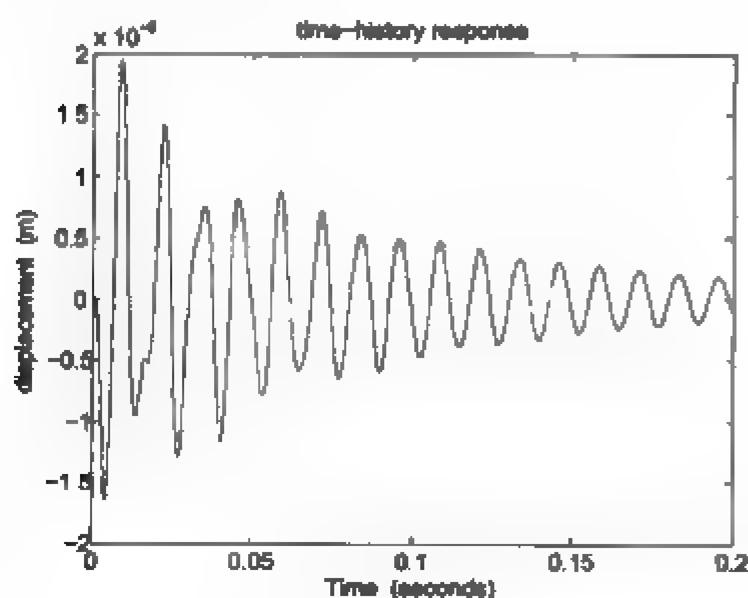
gcoord=[gcoord(:,2:4)];
%-----------------------------------------------------------------------
```

```
%  (0.4) nodal connectivity of the elements
%--         ---------------------------------------------------------
nodes=[  1          1          2;
         2          2          3;
         3          3          4;
         4          4          5;
         5          5          6;
         6          6          7;
         7          7          8;
         8          8          9;
         9          9         10;
        10         10         11;
        11         11         12];

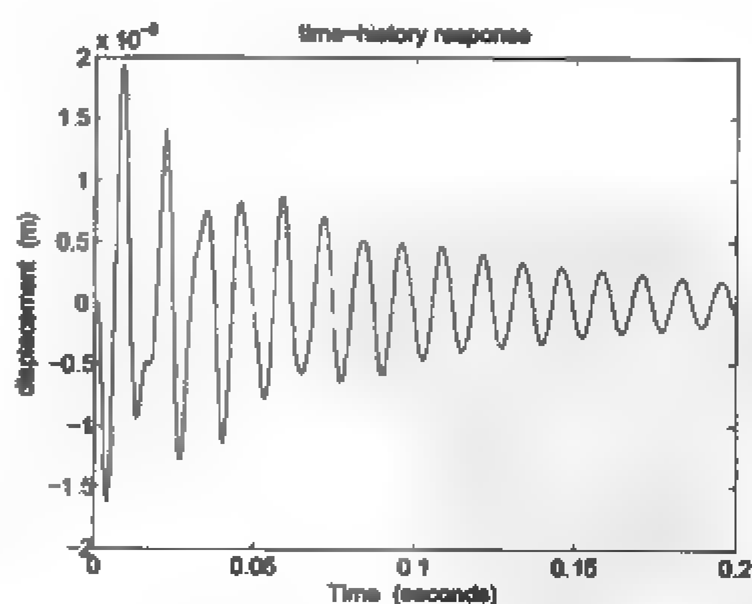
nodes=[nodes(:,2:3)];
%-------------------------------------------------------------------
%  (0.5) applied loads
%-------------------------------------------------------------------
                           % a load applied at node 6 in -y direction
P=[-500,6,2];

%-------------------------------------------------------------------
%  (0.6) boundary conditions
%-------------------------------------------------------------------
ConNode=[ 1,  1,  1,  1,  1,  1,  1;    % code for constrained nodes
         12,  1,  1,  1,  1,  1,  1];
ConVal =[ 1,  0,  0,  0,  0,  0,  0;     % values at constrained dofs
         12,  0,  0,  0,  0,  0,  0];
%-------------------------------------------------------------------
%    The end
%-------------------------------------------------------------------
```

用 Euler Bernoulli 梁单元(集中质量)进行计算，结构在节点 7 沿 y 方向的响应如图 5.11 所示.

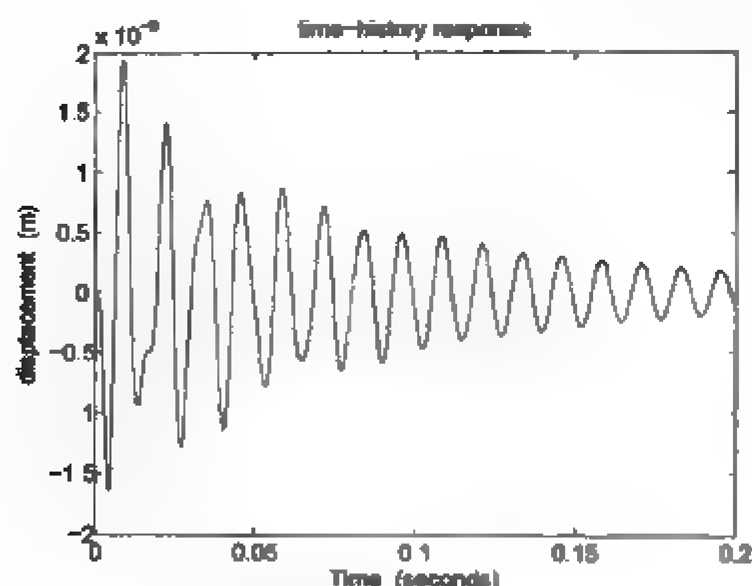


(a) 中心差分法计算的响应

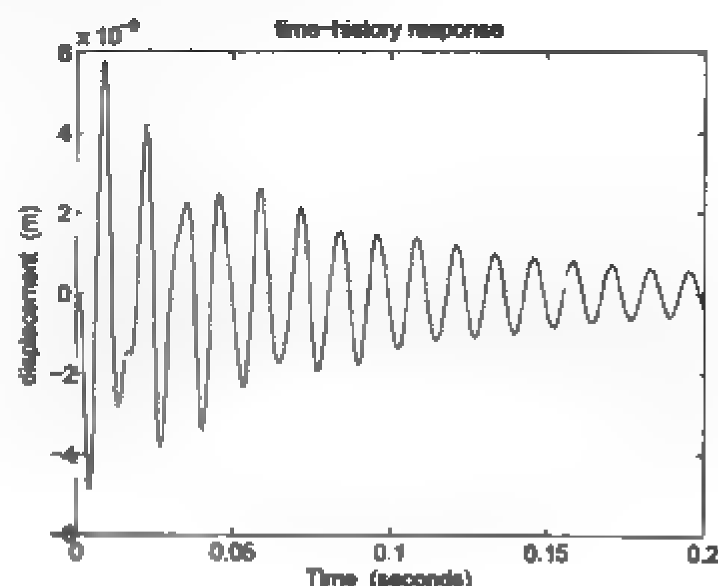


(b) Houbolt 法计算的响应

图 5.11　直接积分法计算的刚架结构节点 7 处的响应



(c) Wilson-θ 法计算的响应



(d) Newmark 法计算的响应

图 5.11 (续)

5.4.4 频响分析

在一些实际应用中，如主动控制、模态试验，前述的时域方法并不方便，这时可以应用频域方法进行分析.

结构系统的动态特性也可以由其本身的固有频率成分来表征，它们决定了系统的时间和频率响应. 进行频域响应计算的常用工具是 FFT(Fast Fourier Transform).

对于连续时间函数 $x(t)$，可以表示为

$$x(t)=\frac{1}{2\pi}\int_{-\infty}^{\infty}X(\omega)\mathrm{e}^{\mathrm{j}\omega t}\mathrm{d}\omega \tag{5-119}$$

其中

$$X(\omega)=\int_{-\infty}^{\infty}x(t)\mathrm{e}^{-\mathrm{j}\omega t}\mathrm{d}t \tag{5-120}$$

$X(\omega)$ 称为时域函数 $x(t)$ 的 Fourier 变换，而 $x(t)$ 则称为 $X(\omega)$ 的 Fourier 反变换.

引入变量 $f=\omega/2\pi$，Fourier 变换可表示为

$$x(t)=\int_{-\infty}^{\infty}X(f)\mathrm{e}^{\mathrm{j}(2\pi f)t}\mathrm{d}f \tag{5-121}$$

和

$$X(f)=\int_{-\infty}^{\infty}x(t)\mathrm{e}^{-\mathrm{j}(2\pi f)t}\mathrm{d}t \tag{5-122}$$

对于上述的积分计算可以采用数值积分的方法进行计算.

设有连续函数 $x(t)$ 经过 N 个采样后的函数序列

$$x_k=x(t_k),\quad t_k=k\Delta t\quad (k=0,1,2,\cdots,N-1) \tag{5-123}$$

相应地在频域上映射有 N 个频率

$$f_0,f_2,\cdots,f_{N-1} \tag{5-124}$$

经过离散化，有

$$X(f_n)=\int_{-\infty}^{\infty}x(t)\mathrm{e}^{-\mathrm{j}2\pi f_n t}\mathrm{d}t\approx\sum_{k=0}^{N-1}x(t_k)\mathrm{e}^{-\mathrm{j}2\pi f_n t_k}\Delta t \tag{5-125}$$

由于

$$\Delta t=\frac{T}{N},\quad t_k=k\Delta t,\quad f_n=\frac{n}{T}$$

于是

$$X(f_n)=\Delta t\sum_{k=0}^{N-1}x(k\Delta t)\mathrm{e}^{-\mathrm{j}(2\pi/N)nk}\quad(n=0,1,2,\cdots,N-1) \tag{5-126}$$

离散 Fourier 变换(DFT)和离散 Fourier 反变换(IDFT)定义为

$$X(n)=\sum_{k=0}^{N-1}x(k)\mathrm{e}^{-\mathrm{j}(2\pi/N)nk}\quad(n=0,1,2,\cdots,N-1) \tag{5-127}$$

$$x(k)=\frac{1}{N}\sum_{n=0}^{N-1}X(n)\mathrm{e}^{\mathrm{j}(2\pi/N)nk}\quad(n=0,1,2,\cdots,N-1) \tag{5-128}$$

在 MATLAB 中，DFT 的表达式为

$$X(n)=\sum_{k=1}^{N}x(k)\mathrm{e}^{-\mathrm{j}(2\pi/N)(n-1)(k-1)}\ (n=1,2,\cdots,N) \tag{5-129}$$

$$x(k)=\frac{1}{N}\sum_{n=1}^{N}X(n)\mathrm{e}^{\mathrm{j}(2\pi/N)(n-1)(k-1)}\quad(n=1,2,\cdots,N) \tag{5-130}$$

离散 Fourier 变换关于频率 f_n 具有对称性

$$X(f_n)=X(f_{N-n}),\quad f_n=\frac{n}{N\Delta t}\ \left(n=-\frac{N}{2},\cdots,\frac{N}{2}\right) \tag{5-131}$$

因而只需要计算一半频谱. 最大的频率为

$$0<f<\frac{1}{2\Delta t} \tag{5-132}$$

FFT(Fast Fourier Transform)是改进的 DFT，它可以有效地改进计算效率.

【例 5.8】 对例 5.7 中的平面刚架结构的时域响应信号，进行频域响应计算. 时域瞬态响应计算与例 5.7 相同，频域响应计算利用 MATLAB 中的 fft 函数计算，其中主要用到的 MATLAB 函数(见 5.5 节)有：

- FrameElement31(prop,leng,beta,xi,al,Opt_section,Opt_mass)——用 Euler Bernoulli 梁计算单元的刚度矩阵 k 和质量矩阵 m.
- FrameElement32(prop,leng,beta,xi,al,Opt_section,Opt_mass) ——用 Timoshenko 梁计算单元的刚度矩阵 k 和质量矩阵 m.
- femEldof(nd,No_nel,No_dof)——计算单元自由度对应到结构系统上的自由度.
- femAssemble1(kk,k,index)——用于将单元刚度矩阵或质量矩阵组集到系统的总体刚度矩阵 kk 或总体质量矩阵 mm.
- femApplybc1(kk,mm,ff,bcdof,bcval)——对结构系统施加边界条件.
- kkCheck1(kk,mm,ff,bcdof,bcval)——检查总体刚度矩阵 kk 的主元是否为零.
- mmCheck1(kk0,mm0,ff0,bcdof0,bcval0)——检查总体质量矩阵 mm 的主元是否为零.

- bcCheck1(kk0,mm0,ff0,bcdof0,bcval0)——检查边界条件，消去相应的自由度.
- TransResp1(kk1,cc1,mm1,ft0,bcdof1,nt,dt,q0,dq0)——中心差分法计算响应.
- TransResp3(kk1,cc1,mm1,ft0,bcdof1,nt,dt,q0,dq0)——Houbolt 法计算响应.
- TransResp4(kk1,cc1,mm1,ft0,bcdof1,nt,dt,q0,dq0)——Wilson-θ 法计算响应.
- TransResp5(kk1,cc1,mm1,ft0,bcdof1,nt,dt,q0,dq0)——Newmark 法计算响应.

femFFT (yt,tt) ——利用fft函数计算频域响应.

M 文件如下：

```
%----------------------------------------------------------------------------%
% Example 5.8
%   To solve transient response of a 2-d truss structure and
%   calculate Fast Fourier Transform (FFT) of the time domain response.
%   The solution methods are: 1) central difference scheme. 3) Houbolt
%   integration scheme.
%   4) Wilson θ integration scheme. 5) Newmark integration scheme
%   nodal dof: {u1  v1  w1 θx1 θy1 θz1  u2  v2  w2 θx2 θy2 θz2}
% Problem description
%   Find the response of a frame structure which is made of three beams
%   of lengths of 4 m, 3 m and 4 m, respectively. All beams have cross-
%   section of 0.10 m height by 0.05 m width. The elastic modulus is
%   2.10x10^11 Pa. The frame is subjected to an impulse load of amplitude
%   500 N in the middle of the top beam. One end of the each vertical
%   beam is fixed. (see Fig. 5-9 for the element discretization)
% Variable descriptions
%   k, m - element stiffness matrix and mass matrix
%   kk, mm - system stiffness matrix and mass matrix
%   ff - system force vector
%   index - a vector containing system dofs associated with each element
%   bcdof - a vector containing dofs associated with boundary conditions
%   bcval - a vector containing boundary condition values associated
%   with the dofs in 'bcdof'
%   dsp - displacement matrix
%   vel - velocity matrix
%   acc - acceleartion matrix
%----------------------------------------------------------------------------
%  (0) input control data
%----------------------------------------------------------------------------
clear; clc;

Beam_InputData547;      % import the input data for the information of
                       % nodes, elements, loads, constraints and materials
Opt_beam=1;                              % option for type of the beam
                                               % =1 Euler Bernoulli beam
                                                 % =2 Timoshenko beam
Opt_mass=2;                                        % option for mass matrix
```

```
                                        % =1 consistent mass matrix
                                          % =2 lumped mass matrix
Opt_section=1;                     % option for type of cross-section
                                   % = 1 rectangular cross-section
                                       % = 2 circular cross-section
TypeMethod=4;              % option for selecting the solution method
                                   % = 1 central difference scheme
                                   % = 3 Houbolt integration scheme
                                   % = 4 Wilson θ integration scheme
                                   % = 5 Newmark integration scheme
Typeload=1;                        % option for selecting the load type
                                              % = 1 impulse load
                                              % = 2 step load
                                              % = 3 Harmonic load

dt=0.0005;                            % time step size
ti=0;                                 % initial time
tf=0.200;                               % final time
nt=fix((tf-ti)/dt);                   % number of time steps
tt=ti:dt:ti+nt*dt;                 % generate time samples vector

ac=0.00002; bc=0.00008;    % Parameters for proportional damping

al=0;              % angle between the reference coordinate system and
                   % the local coordinate system for the space element
%---------------------------------------------------------------------
%  (7) calculate transient response
%---------------------------------------------------------------------
[kk1,mm1,ff1,bcdof1,bcval1,sdof1]=mmCheck1(kk0,mm0,ff0,bcdof0,bcval0);
      % check the zero main elements in mm and eliminate the rows
      % and columns in equation associated with the zero main elements
switch Typeload
  case 1                              % Impulse force function
    u=[1,zeros(1,nt)];
    ft0=ff1*u;
  case 2                              % Step force function
    u(1,1:nt+1)=1;
    ft0=ff1*u;
  case 3                              % Harmonic force function
    u=cos(omega0*tt);
    ft0=ff1*u;
  otherwise
    ft0=ff1;                          % a given force function
end

cc1=ac*mm1+bc*kk1;                    % Form the Rayleigh damping matrix
q0=zeros(sdof1,1); dq0=zeros(sdof1,1);
```

```
% initial displacement and velocity

switch TypeMethod
  case 1                              % central difference scheme
    [acc,vel,dsp]=TransResp1(kk1,cc1,mm1,ft0,bcdof1,nt,dt,q0,dq0);
  case 3                              % Houbolt integration scheme
    [acc,vel,dsp]=TransResp3(kk1,cc1,mm1,ft0,bcdof1,nt,dt,q0,dq0);
  case 4                                        % Wilson θ integration scheme
    [acc,vel,dsp]=TransResp4(kk1,cc1,mm1,ft0,bcdof1,nt,dt,q0,dq0);
  case 5                                          % Newmark integration scheme
    [acc,vel,dsp]=TransResp5(kk1,cc1,mm1,ft0,bcdof1,nt,dt,q0,dq0);
  otherwise
    disp('Unknown method.')
end
%------------------------------------------------------------------------
%  (8) Calculate FFT of the time domain data y(n,:)
%------------------------------------------------------------------------
jth=20;

yt=dsp(jth,:)';      % extract one time domain data from the response
tt=ti:dt:ti+nt*dt;                        % generate time vector

[yfft, freq]=femFFT(yt,tt);  %  Calculate FFT of the time domain data and
                             %  take absolute values of the result
%------------------------------------------------------------------------
%  (9) graphics of dynamic response
%------------------------------------------------------------------------
jth=20;

figure(1)
   % Plot the graph of the time-history response at jth degree of freedom
plot(tt,dsp(jth,:))
xlabel('Time  (seconds)'), ylabel('displacement  (m)')
title('time-history response')

figure(2)                % Plot the graph of the FFT versus frequency
plot(freq,yfft)
xlabel('Frequency (rad/s)')
ylabel('Absolute values of FFT')
title('FFT result')
%------------------------------------------------------------------------
%  (10) print fem solutions
%------------------------------------------------------------------------
switch Typeload
  case 1
    disp('The excitation is impulse force')
  case 2
```

```
    disp('The excitation is step force')
  case 3
    disp('The excitation is step force')
    otherwise
    disp('The given foece')
end

disp('The calculation is use of:')

if Opt_beam==1
  disp('Euler-Bernoulli beam element 1')
elseif Opt_beam==2
  disp('Timoshenko beam element')
else
  disp('Euler-Bernoulli beam element 2')
end

if Opt_mass==1
  disp('and consistent mass matrix')
elseif Opt_mass==2
  disp('and lumped mass matrix')
else
  disp('and diagonal mass matrix')
end

num=1:1:sdof2;
frequency=[num' omega1]                          % print natural frequency
%------------------------------------------------------------------------
%     The end
%------------------------------------------------------------------------
```

平面刚架结构节点 7 处 y 方向的响应信号的频域响应如图 5.12 所示.

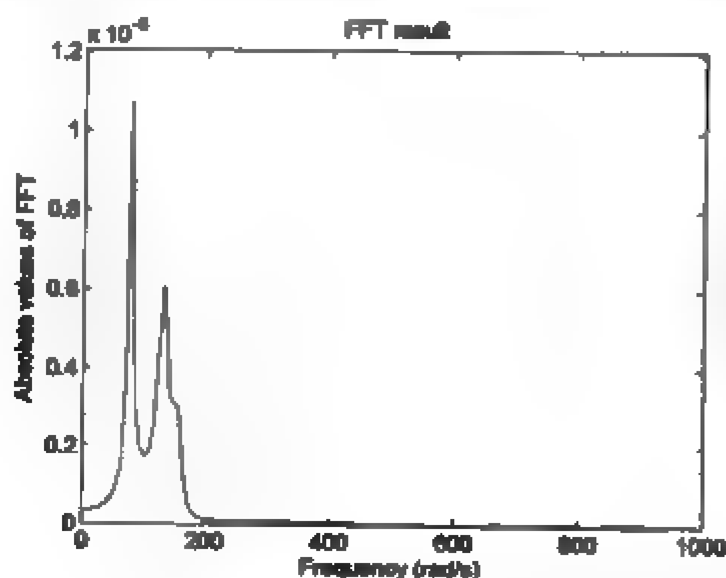


图 5.12　平面刚架结构节点 7 处的频域响应

5.5 应用问题的 MATLAB 函数

1. femEldof 函数

femEldof 函数的代码如下.

```
%-------------------------------------------------------------------------
function [index]=femEldof(nd,No_nel,No_dof)
%-------------------------------------------------------------------------
% Purpose:
%    Compute system dofs associated with each element
% Synopsis:
%    [index]=femEldof(nd,No_nel,No_dof)
% Variable Description:
%    index - system dof vector associated with element "iel"
%    nd - node connectivity for the (iel)-th element
%    No_nel - number of nodes per element
%    No_dof - number of dofs per node
%    iel - element number whose system dofs are to be determined
%-------------------------------------------------------------------------
  k=0;
  for i=1:No_nel
    start = (nd(i)-1)*No_dof;
      for j=1:No_dof
        k=k+1;
        index(k)=start+j;
      end
  end
%-------------------------------------------------------------------------
%    The end
%-------------------------------------------------------------------------
```

2. femAssemble1 函数

femAssemble1 函数的代码如下.

```
%-------------------------------------------------------------------------
function [kk]=femAssemble1(kk,k,index)
%-------------------------------------------------------------------------
% Purpose:
%    Assembly of element matrices into the system matrix
% Synopsis:
%    [kk]=femAssemble1(kk,k,index)
% Variable Description:
%    kk - system matrix
%    k  - element matrix
%    index - dof vector associated with an element
```

```
%----------      ---------------------------------------------------
eldof = length(index);

for i=1:eldof
   ii=index(i);
   for j=1:eldof
      jj=index(j);
      kk(ii,jj)=kk(ii,jj)+k(i,j);
   end
end
%-------------------------------------------------------------------
%    The end
%-------------------------------------------------------------------
```

3. femApplybc1 函数

femApplybc1 函数的代码如下.

```
%-------------------------------------------------------------------
function [kk,mm,ff]=femApplybc1(kk,mm,ff,bcdof,bcval)
%-------------------------------------------------------------------
% Purpose:
%    Apply constraints to matrix equation [kk]{x}={ff}
%    Apply constraints to eigenvalue matrix equation
%    [kk]{x}=lamda[mm]{x}
% Synopsis:
%    [kk,mm,ff]=femApplybc1(kk,mm,ff,bcdof,bcval)
% Variable Description:
%    kk - system stiffness matrix before applying constraints
%    mm - system mass matrix before applying constraints
%    ff - system vector before applying constraints
%    bcdof - a vector containing constrained dof
%    bcval - a vector containing contained value
%-------------------------------------------------------------------
ni=length(bcdof);
sdof=size(kk);

for ii=1:ni
  if bcdof(ii)==1
    for j=1:sdof
      kk(ii,j)=0;
      kk(j,ii)=0;
      mm(ii,j)=0;
      mm(j,ii)=0;
      ff(j)=ff(j)-bcval(ii)*kk(j,ii);
    end
    kk(ii,ii)=1;
    ff(ii)=bcval(ii);
```

```
  end
 end
%-----------------------------------------------------------------------
%    The end
%-----------------------------------------------------------------------
```

4. kkCheck1 函数

kkCheck1 函数的代码如下.

```
%-----------------------------------------------------------------------
function [kk1,mm1,ff1,bcdof1,bcval1,sdof1]=kkCheck1(kk,mm,ff,bcdof,bcval)
%-----------------------------------------------------------------------
%  Purpose:
%    Check whether the main elements of system stiffness matrix are
%    zeros and eliminate the columns and rows associated with the zero
%    main element in the system matrix equation.
%  Synopsis:
%    [kk1,mm1,ff1,bcdof1,bcval1]=kkcheck1(kk,mm,ff,bcdof,bcval)
%  Variable Description:
%    kk = system stiffness matrix
%    mm = system mass vector
%    ff = system force vector
%    sdof = total degrees of freedom of the system
%    bcdof = a vector containing dofs associated with boundary conditions
%    bcval = a vector containing boundary condition values associated
%    with the dofs in 'bcdof'
%-----------------------------------------------------------------------
%  (1) check the zero main element in the kk
%-----------------------------------------------------------------------
[sdof,n1]=size(kk);

jk=0;

for ii=1:sdof          % loop for check of the zero main elements in kk
  check=kk(ii,ii);
  if check==0
    jk=jk+1;                  % location of the zero main element
    kki(jk)=ii;         % storring the location of the zero main element
  end
end

if jk~=0
  disp('main elements of kk:')        % display the zero main element
  kkii=kki
  disp('equal zero')
end
%-----------------------------------------------------------------------
```

```
% (2) eliminate the columns and rows in kk, mm, ff, bcdof, bcval
%------------------------------------------------------------------------
if jk~=0
  for jj=jk:-1:1  % loop for moving the columns and rows associated with
                  % the zero main element in the system matrix equation
    hh=sdof-kki(jj);
    hr=kki(jj);

    for i=1:hh                % exchanging the rows in the equation
      kt=kk(hr+i-1,:);
      mt=mm(hr+i-1,:);
      ft=ff(hr+i-1);
      bdt=bcdof(hr+i-1);
      bvt=bcval(hr+i-1,:);

      kk(hr+i-1,:)=kk(hr+i,:);
      mm(hr+i-1,:)=mm(hr+i,:);
      ff(hr+i-1)=ff(hr+i);
      bcdof(hr+i-1)=bcdof(hr+i);
      bcval(hr+i-1,:)=bcval(hr+i,:);

      kk(hr+i,:)=kt;
      mm(hr+i,:)=mt;
      ff(hr+i)=ft;
      bcdof(hr+i,:)=bdt;
      bcval(hr+i,:)=bvt;
    end

    for j=1:hh          % exchanging the columns in the equation
      kt=kk(:,hr+j-1);
      mt=mm(:,hr+j-1);
      kk(:,hr+j-1)=kk(:,hr+j);
      mm(:,hr+j-1)=mm(:,hr+j);
      kk(:,hr+j)=kt;
      mm(:,hr+j)=mt;
    end

  end

end

sdof1=sdof-jk;
kk1=[kk(1:sdof1,1:sdof1)];  % eliminating the columns and rows of the kk
mm1=[mm(1:sdof1,1:sdof1)];  % eliminating the columns and rows of the mm
ff1=[ff(1:sdof1)];                        % eliminating the rows of the ff
bcdof1=[bcdof(1:sdof1)];              % eliminating the rows of the bcdof
bcval1=[bcval(1:sdof1,:)];            % eliminating the rows of the bcval
```

```
%--------------------------------------------------------------------
%    The end
%--------------------------------------------------------------------
```

5. mmCheck1 函数

mmCheck1 函数的代码如下.

```
%--------------------------------------------------------------------
function [kk2,mm2,ff2,bcdof2,bcval2,sdof2]=mmCheck1(kk,mm,ff,bcdof,bcval)
%--------------------------------------------------------------------
% Purpose:
%    Check whether the main elements of system mass matrix are zeros and
%    eliminate the columns and rows associated with the zero main
%    element in the system matrices kk, mm, ff, bcdof, bcval
% Synopsis:
%    [kk2,mm2,ff2,bcdof2,bcval2,sdof2]=mmCheck1(kk,mm,ff,bcdof,bcval)
% Variable Description:
%    kk, mm = system stiffness matrix and system mass matrix
%    ff = system force vector
%    bcdof = a vector containing dofs associated with boundary conditions
%    bcval = a vector containing boundary condition values associated
%    with the dofs in 'bcdof'
%    sdof = total degrees of freedom of the system
%--------------------------------------------------------------------
% (1) check the zero main element in the mm
%--------------------------------------------------------------------
[sdof,n2]=size(kk);

jk=0;

for ii=1:sdof          % loop for check of the zero main elements in mm
  check=mm(ii,ii);
  if check==0
    jk=jk+1;               % location of the zero main element
    mmi(jk)=ii;        % storing the location of the zero main element
  end
end
%--------------------------------------------------------------------
% (2) eliminate the columns and rows in kk, mm, ff, bcdof, bcval
%--------------------------------------------------------------------
if jk~=0
  for jj=jk:-1:1   % loop for moving the columns and rows associated with
                   % the zero main element in the system matrix equation
    hh=sdof-mmi(jj);
    hr=mmi(jj);

    for i=1:hh              % exchanging the rows in the equation
```

```
      kt=kk(hr+i-1,:);
      mt=mm(hr+i-1,:);
      ft=ff(hr+i-1);
      bdt=bcdof(hr+i-1);
      bvt=bcval(hr+i-1);
      kk(hr+i-1,:)=kk(hr+i,:);
      mm(hr+i-1,:)=mm(hr+i,:);
      ff(hr+i-1)=ff(hr+i);
      bcdof(hr+i-1)=bcdof(hr+i);
      bcval(hr+i-1)=bcval(hr+i);
      kk(hr+i,:)=kt;
      mm(hr+i,:)=mt;
      ff(hr+i)=ft;
      bcdof(hr+i)=bdt;
      bcval(hr+i)=bvt;
    end

    for j=1:hh              % exchanging the columns in the equation
      kt=kk(:,hr+j-1);
      mt=mm(:,hr+j-1);
      kk(:,hr+j-1)=kk(:,hr+j);
      mm(:,hr+j-1)=mm(:,hr+j);
      kk(:,hr+j)=kt;
      mm(:,hr+j)=mt;
    end

  end

end

sdof2=sdof-jk;
kk2=[kk(1:sdof2,1:sdof2)];   % eliminating the columns and rows of the kk
mm2=[mm(1:sdof2,1:sdof2)];   % eliminating the columns and rows of the mm
ff2=[ff(1:sdof2)];               % eliminating the rows of the ff
bcdof2=[bcdof(1:sdof2)];       % eliminating the rows of the bcdof
bcval2=[bcval(1:sdof2)];        % eliminating the rows of the bcval
%--------------------------------------------------------------------------
%    The end
%--------------------------------------------------------------------------
```

6. bcCheck1 函数

bcCheck1 函数的代码如下.

```
%--------------------------------------------------------------------------
function [kk2,mm2,ff2,sdof2]=bcCheck1(kk,mm,ff,bcdof,bcval)
%--------------------------------------------------------------------------
%  Purpose:
```

```
%    Check the boundary conditions and eliminate the columns and rows
%    associated with the boundary conditions in the system equation
% Synopsis:
%    [kk2,mm2,ff2,sdof2]=bcChec1k(kk,mm,ff,,bcdof,bcval)
% Variable Description:
%    kk, mm = system stiffness matrix and system mass matrix
%    ff = system force vector
%    bcdof = a vector containing dofs associated with boundary conditions
%    bcval = a vector containing boundary condition values associated
%    with the dofs in 'bcdof'
%-----------------------------------------------------------------------
% (1) check the boundary conditions
%-----------------------------------------------------------------------
[sdof,n1]=size(kk);

jk=0;

for ii=1:sdof              % loop for check of the boundary conditions

  check=bcdof(ii);
  if check==1
    jk=jk+1;               % location of the zero main element
    bci(jk)=ii;            % storing the location of the zero main element
  end

end
%-----------------------------------------------------------------------
% (2) eliminate the columns and rows associated with bcs
%-----------------------------------------------------------------------
if jk~=0

  for jj=jk:-1:1                  % loop for moving the columns and rows
                                 % associated with boundary conditions
    hh=sdof-bci(jj);
    hr=bci(jj);

    for i=1:hh                   % exchanging the columns of the kk, etc.
      kt=kk(hr+i-1,:);
      mt=mm(hr+i-1,:);
      ft=ff(hr+i-1);
      kk(hr+i-1,:)=kk(hr+i,:);
      mm(hr+i-1,:)=mm(hr+i,:);
      ff(hr+i-1)=ff(hr+i);
      kk(hr+i,:)=kt;
      mm(hr+i,:)=mt;
      ff(hr+i)=ft;
    end
```

```
    for j=1:hh                % exchanging the rows of the kk, mm, ff
      kt=kk(:,hr+j-1);
      mt=mm(:,hr+j-1);
      kk(:,hr+j-1)=kk(:,hr+j);
      mm(:,hr+j-1)=mm(:,hr+j);
      kk(:,hr+j)=kt;
      mm(:,hr+j)=mt;
    end

  end

end

sdof2=sdof-jk;
kk2=[kk(1:sdof2,1:sdof2)];  % eliminating the columns and rows of the kk
mm2=[mm(1:sdof2,1:sdof2)];  % eliminating the columns and rows of the mm
ff2=[ff(1:sdof2)];                  % eliminating the rows of the ff
%----------------------------------------------------------------------
%    The end
%----------------------------------------------------------------------
```

7. BeamElement11 函数

BeamElement11 函数的代码如下.

```
%----------------------------------------------------------------------
function [k,m]=BeamElement11(prop,leng,Opt_mass)
%----------------------------------------------------------------------
% Purpose:
%    To calculate stiffness and mass matrices for the beam element.
%    (Euler-Bernoulli beam)
%    nodal dof: {v1 θ1  v2 θ2}
% Synopsis:
%    [k,m]=BeamElement11(prop,leng,Opt_mass)
% Variable Description:
%    k, m - element stiffness matrix and element mass matrix
%    prop - the properties of materials and geometry
%    leng - element length
%    Opt_mass  = 1 - consistent mass matrix
%              = 2 - lumped mass matrix
%----------------------------------------------------------------------
% (0) evaluation of the constants
%----------------------------------------------------------------------
   E=prop(1);                                        % elastic modulus
   u=prop(2);                                         % Poisson's ratio
   rho=prop(3);             % mass density (mass per unit volume)
```

```
    A=prop(6);                            % area of beam cross-section
    Iz=prop(8);    % 2nd moment of inertia of cross-section about axis z

    G=E/(2*(1+u));                                       % shear modulus
%-----------------------------------------------------------------------
%  (1) element stiffness matrix
%-----------------------------------------------------------------------
    c=E*Iz/(leng^3);
    k0=[12       6*leng    -12       6*leng;
        6*leng   4*leng^2  -6*leng   2*leng^2;
       -12      -6*leng     12      -6*leng;
        6*leng   2*leng^2  -6*leng   4*leng^2];
    k=c*k0;
%-----------------------------------------------------------------------
%  (2) element mass matrix
%-----------------------------------------------------------------------
if Opt_mass==1
%----------------------------
%  (2.1) consistent mass matrix
%----------------------------
   mass=rho*A*leng;
   m0=[156       22*leng    54        -13*leng;
       22*leng   4*leng^2   13*leng   -3*leng^2;
       54        13*leng    156       -22*leng;
      -13*leng  -3*leng^2  -22*leng    4*leng^2];
   m=mass/420*m0;
elseif Opt_mass==2
%---------------------------
%  (2.2) lumped mass matrix
%---------------------------
    mass=rho*A*leng;
    m0=diag([1  0  1  0]);
    m=mass/2*m0;
end
%-----------------------------------------------------------------------
%    The end
%-----------------------------------------------------------------------
```

8. BeamElement12 函数

BeamElement12 函数的代码如下.

```
%-----------------------------------------------------------------------
function [k,m]=BeamElement12(prop,leng,Opt_mass)
%-----------------------------------------------------------------------
% Purpose:
%    To calculate stiffness and mass matrices for the beam element.
%    (Timoshenko beam)
```

```
%     nodal dof: {v1 θ1  v2 θ2}
%  Synopsis:
%     [k,m]=BeamElement12(prop,leng,Opt_mass)
%  Variable Description:
%     k, m - element stiffness matrix and element mass matrix
%     prop - the properties of materials and geometry
%     leng - element length
%     Opt_mass  = 1 - consistent mass matrix
%               = 2 - lumped mass matrix
%-----------------------------------------------------------------------------
%  (0) evaluation of the constants
%-----------------------------------------------------------------------------
   E=prop(1);                                              % elastic modulus
   u=prop(2);                                              % Poisson's ratio
   rho=prop(3);               %  mass density (mass per unit volume)

   A=prop(6);                              % area of beam cross-section
   Iz=prop(8);     % 2nd moment of inertia of cross-section about axis z

   G=E/(2*(1+u));                                          % shear modulus
%-----------------------------------------------------------------------------
%  (1) element stiffness matrix
%-----------------------------------------------------------------------------
   c=E*Iz/leng;
   d=(5/6)*G*A/(4*leng);
   k=[4*d        2*d*leng    -4*d        2*d*leng;
      2*d*leng   c+d*leng^2  -2*d*leng   -c+d*leng^2;
     -4*d       -2*d*leng     4*d       -2*d*leng;
      2*d*leng  -c+d*leng^2  -2*d*leng   c+d*leng^2];
%-----------------------------------------------------------------------------
%  (2) element mass matrix
%-----------------------------------------------------------------------------
if Opt_mass==1
%---------------------------
%  (2.1) consistent mass matrix
%---------------------------
   mass=rho*A*leng;
   m0=[2    0    1    0;
       0    0    0    0;
       1    0    2    0;
       0    0    0    0];
   m=mass/6*m0;
else
%--------------------------
%  (2.2) lumped mass matrix
%--------------------------
   mass=rho*A*leng;
```

```
    m0=diag([1  0  1  0]);
    m=mass/2*m0;
end
%----------------------------------------------------------------------
%    The end
%----------------------------------------------------------------------
```

9. BeamElement14 函数

BeamElement14 函数的代码如下.

```
%----------------------------------------------------------------------
function [k,m]=BeamElement14(prop,leng,Opt_mass)
%----------------------------------------------------------------------
% Purpose:
%    To calculate stiffness and mass matrices for the beam element.
%    (mixed beam)
%    nodal dof: {M1  v1  M2  v2}
% Synopsis:
%    [k,m]=BeamElement14(prop,leng,Opt_mass)
% Variable Description:
%    k, m - element stiffness matrix and element mass matrix
%    prop - the properties of materials and geometry
%    leng - element length
%    Opt_mass = 1 - consistent mass matrix
%             = 2 - lumped mass matrix
%----------------------------------------------------------------------
% (0) evaluation of the constants
%----------------------------------------------------------------------
    E=prop(1);                                          % elastic modulus
    u=prop(2);                                          % Poisson's ratio
    rho=prop(3);                 %  mass density (mass per unit volume)
    A=prop(6);                   % area of beam cross-section
    Iz=prop(8);     % 2nd moment of inertia of cross-section about axis z
    shmodule=prop(10);                      % selection of shear modulus
%----------------------------------------------------------------------
% (1) element stiffness matrix
%----------------------------------------------------------------------
if shmodule==0
%----------------------------------------------
% (1.1) stiffness matrix not including shear deformation
%----------------------------------------------
    c=1/(6*E*Iz*leng);
    k=[2*leng^2    6*E*Iz     leng^2      -6*E*Iz;
       6*E*Iz      0        -6*E*Iz      0;
       leng^2     -6*E*Iz    2*leng^2    6*E*Iz;
       -6*E*Iz     0         6*E*Iz      0];
    k=c*k;
```

```
elseif shmodule==1
%-----------------------------------------
%  (1.2) stiffness matrix including shear deformation
%-----------------------------------------
   G=E/(2*(1+u));                                        % shear modulus
   c=1/(6*E*Iz*leng);
   d=(6/5)*6*E*Iz/(G*A);
   k=[2*leng^2+d    6*E*Iz     leng^2-d      -6*E*Iz;
      6*E*Iz        0         -6*E*Iz        0;
      leng^2-d     -6*E*Iz     2*leng^2+d    6*E*Iz;
      -6*E*Iz       0          6*E*Iz        0];
   k=c*k;
else
%-----------------------------------------
%  (1.3) stiffness matrix including shear deformation
%-----------------------------------------
   G=prop(10);                                           % shear modulus
   c=1/(6*E*Iz*leng);
   d=(6/5)*6*E*Iz/(G*A);
   k=[2*leng^2+d    6*E*Iz     leng^2-d      -6*E*Iz;
      6*E*Iz        0         -6*E*Iz        0;
      leng^2-d     -6*E*Iz     2*leng^2+d    6*E*Iz;
      -6*E*Iz       0          6*E*Iz        0];
   k=c*k;
end
%----------------------------------------------------------------
%  (2) element mass matrix
%----------------------------------------------------------------
if Opt_mass==1
%-------------------------
%  (2.1) consistent mass matrix
%-------------------------
   mass=rho*A*leng;
   m0=diag([0  1  0  1]);
   m=mass/2*m0;
elseif Opt_mass==2
%-----------------------
%  (2.2) lumped mass matrix
%-----------------------
   mass=rho*A*leng;
   m0=diag([0  1  0  1]);
   m=mass/2*m0;
else
%------------------------
%  (2.3) diagonal mass matrix
%------------------------
   mass=rho*A*leng;
```

```
    m0=diag([1  1  1  1]);
    m=mass/2*m0;
end
%------------------------------------------------------------------------
%    The end
%------------------------------------------------------------------------
```

10. FrameElement21 函数

FrameElement21 函数的代码如下.

```
%------------------------------------------------------------------------
function [k,m]=FrameElement21(prop,leng,beta,Opt_section,Opt_mass)
%------------------------------------------------------------------------
%  Purpose:
%     To calculate stiffness and mass matrices for the 2-d frame element
%     (Euler-Bernoulli beam)
%     nodal dof: {u1  v1  θ1  u2  v2  θ2}
%  Synopsis:
%     [k,m]=FrameElement21(prop,leng,beta,Opt_section,Opt_mass)
%  Variable Description:
%     k, m - element stiffness matrix and element mass matrix
%     prop - the properties of materials and geometry:
%     leng - element length
%     beta - angle between the local and global axes
%     is positive if the local axis is in the ccw direction from the global axis
%     Opt_section - option for type of cross-section
%               = 1 - rectangular cross-section
%               = 2 - circular cross-section
%     Opt_mass - option for mass matrix
%             = 1 - consistent mass matrix
%             = 2 - lumped mass matrix
%             = 3 - diagonal mass matrix
%------------------------------------------------------------------------
%  (0) evaluation of the constants
%------------------------------------------------------------------------
E=prop(1);                                         % elastic modulus
u=prop(2);                                          % Poisson's ratio
rho=prop(3);                     %  mass density (mass per unit volume)

if Opt_section==1
  h=prop(4);                        % height of beam cross-section
  b=prop(5);                        % width of beam cross-section
elseif Opt_section==2
  D=prop(4);                     % outer diameter of beam cross-section
  d=prop(5);                     % inner diameter of beam cross-section
end
```

```
A=prop(6);                              % area of beam cross-section
Iy=prop(7);         % 2nd moment of inertia of cross-section about axis y
Iz=prop(8);         % 2nd moment of inertia of cross-section about axis z
%-------------------------------------------------------------------------
%  (1) rotation matrix for the coordinate transformation
%-------------------------------------------------------------------------
cc=cos(beta); ss=sin(beta);

 T=[ cc   ss    0    0    0    0;
    -ss   cc    0    0    0    0;
     0    0    1    0    0    0;
     0    0    0   cc   ss    0;
     0    0    0  -ss   cc    0;
     0    0    0    0    0    1];
%-------------------------------------------------------------------------
%  (2) stiffness matrix
%-------------------------------------------------------------------------
%-----------------------------------
%  (2.1) stiffness matrix at the local axis
%-----------------------------------
 ka=E*A/leng; kc=E*Iz/(leng^3);

 k0=[ka   0           0            -ka     0           0;
      0   12*kc       6*leng*kc      0    -12* kc      6*leng*kc;
      0   6*leng*kc   4*leng^2*kc    0    -6*leng*kc   2*leng^2*kc;
    -ka   0           0             ka     0           0;
      0  -12*kc      -6*leng*kc      0     12*kc      -6*leng*kc;
      0   6*leng*kc   2*leng^2*kc    0    -6*leng*kc   4*leng^2*kc];
%-----------------------------------
%  (2.2) stiffness matrix at the global axis
%-----------------------------------
 k=T'*k0*T;
%-------------------------------------------------------------------------
%  (3) mass matrix
%-------------------------------------------------------------------------
if Opt_mass==1
%---------------------------
%  (3.1) consistent mass matrix
%---------------------------
 ma=rho*A*leng/6; mb=rho*A*leng/420;

 m0=[2*ma  0            0            ma    0             0;
     0     156*mb       22*leng*mb   0     54*mb        -13*leng*mb;
     0     22*leng*mb   4*leng^2*mb  0     13*leng*mb   -3*leng^2*mb;
     ma    0            0            2*ma  0             0;
     0     54*mb        13*leng*mb   0     156*mb       -22*leng*mb;
     0    -13*leng*mb  -3*leng^2*mb  0    -22*leng*mb    4*leng^2*mb];
```

```
elseif Opt_mass==2
%-------------------------
%  (3.2) lumped mass matrix
%-------------------------
  mass=rho*A*leng;
  m0=mass/2*diag([1  1  0  1  1  0]);
end
%-----------------------------------
%  (3.3) mass matrix in the global system
%-----------------------------------
m=T'*m0*T;
%--------------------------------------------------------------------------
%    The end
%--------------------------------------------------------------------------
```

11. FrameElement22 函数

FrameElement22 函数的代码如下.

```
%--------------------------------------------------------------------------
function [k,m]=FrameElement22(prop,leng,beta,Opt_section,Opt_mass)
%--------------------------------------------------------------------------
% Purpose:
%    To calculate stiffness and mass matrices for the 2-d frame element
%    (Timoshenko beam)
%    nodal dof: {u1  v1  θ1  u2  v2  θ2}
% Synopsis:
%    [k,m]=FrameElement22(prop,leng,beta,Opt_section,Opt_mass)
% Variable Description:
%    k, m - element stiffness matrix and element mass matrix
%    prop - the properties of materials and geometry
%    leng - element length
%    beta - angle between the local and global axes
%    is positive if the local axis is in the ccw direction from the global axis
%    Opt_section - option for type of cross-section
%             = 1 - rectangular cross-section
%             = 2 - circular cross-section
%    Opt mass - option for mass matrix
%           = 1 - consistent mass matrix
%           = 2 - lumped mass matrix
%--------------------------------------------------------------------------
% (0) calculation of the constants
%--------------------------------------------------------------------------
%-------------------------------
% (0.1) evaluation of the constants
%-------------------------------
E=prop(1);                              % elastic modulus
u=prop(2);                              % Poisson's ratio
```

```
rho=prop(3);             % mass density (mass per unit volume)

if Opt_section==1
  h=prop(4);                 % height of beam cross-section
  b=prop(5);                 % width of beam cross-section
elseif Opt_section==2
  D=prop(4);             % outer diameter of beam cross-section
  d=prop(5);              % inner diameter of beam cross-section
end

A=prop(6);                          % area of beam cross-section
Iy=prop(7);     % 2nd moment of inertia of cross-section about axis y
Iz=prop(8);     % 2nd moment of inertia of cross-section about axis z

polrmoment=prop(9);       % selection of the polar moment of inertia
shmodule=prop(10);  .             % selection of shear modulus
%--------------------------------
%  (0.2) calculate the shear modulus
%--------------------------------
 if shmodule==0
   G=0;
 elseif shmodule==1
   G=E/(2*(1+u));
 else
   G=prop(10);
 end
%-----------------------------------------------
%  (0.3) selection of correction factor for shear energy
%-----------------------------------------------
if Opt_section==1
  ck=6/5;          % correction factor of the rectangular cross-section
elseif Opt_section==2
  ck=10/9;          % correction factor of the circular cross-section
end
%-------------------------------------------------------------------------
%  (1) rotation matrix for the coordinate transformation
%-------------------------------------------------------------------------
cc=cos(beta); ss=sin(beta);

T=[ cc   ss    0    0    0    0;
   -ss   cc    0    0    0    0;
    0    0    1    0    0    0;
    0    0    0   cc   ss    0;
    0    0    0  -ss   cc    0;
    0    0    0    0    0    1];
%-------------------------------------------------------------------------
%  (2) stiffness matrix
```

```
%------------------------------------------------------------------------
%----------------------------------
%  (2.1) stiffness matrix at the local axis
%----------------------------------
 ka=E*A/leng;
 kc=E*Iz/leng;
 kd=G*A/(4*ck*leng);

 k0=[ka    0         0           -ka     0          0;
     0   4*kd       2*kd*leng      0    -4*kd       2*kd*leng;
     0   2*kd*leng   kc+kd*leng^2   0    -2*kd*leng   -kc+kd*leng^2;
   -ka    0          0             ka    0           0;
     0  -4*kd       -2*kd*leng      0     4*kd       -2*kd*leng;
     0   2*kd*leng  -kc+kd*leng^2   0     -2*kd*leng   kc+kd*leng^2];
%------------------------------------
%  (2.2) stiffness matrix at the global axis
%------------------------------------
k=T'*k0*T;
%------------------------------------------------------------------------
%  (3) mass matrix
%------------------------------------------------------------------------
if Opt_mass==1
%---------------------------
%  (3.1) consistent mass matrix
%---------------------------
   mass=rho*A*leng;
   m0=mass/6*[2     0     0     1     0     0;
              0     2     0     0     1     0;
              0     0     0     0     0     0;
              1     0     0     2     0     0;
              0     1     0     0     2     0;
              0     0     0     0     0     0];
else
%-------------------------
%  (3.2) lumped mass matrix
%-------------------------
   mass=rho*A*leng;
   m0=mass/2*diag([1, 1, 0, 1, 1, 0]);
end
%------------------------------
%  (3.3) mass in the global system
%------------------------------
   m=T'*m0*T;
%------------------------------------------------------------------------
%    The end
% ----------------------------------------------------------------------
```

12. FrameElement31 函数

FrameElement31 函数的代码如下.

```
%-------------------------------------------------------------------------
function [k,m]=FrameElement31(prop,leng,xi,al,Opt_section,Opt_mass)
%-------------------------------------------------------------------------
% Purpose:
%    To calculate stiffness and mass matrices for the 3-d frame element
%    (Euler-Bernoulli beam)
%    nodal dof: {u1  v1  w1 θx1  θy1  θz1  u2  v2  w2  θx2  θy2  θz2}
% Synopsis:
%    [k,m]=FrameElement31(prop,leng,xi,al,Opt_section,Opt_mass)
% Variable Description:
%    k, m - element stiffness matrix and element mass matrix
%    prop - the properties of materials and geometry:
%    leng - element length
%    xi - the first row of the coordinate transform matrix between the
%    local and global axes
%      xi(1)=cos(x,x'), xi(2)=cos(x,y'), xi(3)=cos(x,z')
%    al - angle between the reference coordinate system and the local
%    coordinate system
%      for the space element
%    Opt_section - option for type of cross-section
%             = 1 - rectangular cross-section
%             = 2 - circular cross-section
%    Opt_mass - option for mass matrix
%            = 1 - consistent mass matrix
%            = 2 - lumped mass matrix
%-------------------------------------------------------------------------
% (0) calculation of the constants
%-------------------------------------------------------------------------
%-------------------------------
% (0.1) evaluation of the constants
%-------------------------------
E=prop(1);                                % elastic modulus
u=prop(2);                                 % Poisson's ratio
rho=prop(3);           % mass density (mass per unit volume)
if Opt_section==1
  h=prop(4);                       % height of beam cross-section
  b=prop(5);                        % width of beam cross-section
elseif Opt_section==2
  D=prop(4);            % outer diameter of beam cross-section
  d=prop(5);            % inner diameter of beam cross-section
end
A=prop(6);                              % area of beam cross-section
Iy=prop(7);       % 2nd moment of inertia of cross-section about axis y
```

```
Iz=prop(8);        % 2nd moment of inertia of cross-section about axis z

polrmoment=prop(9);       % selection of the polar moment of inertia
shmodule=prop(10);                 % selection of shear modulus
%------- -  -- ----------------------
%  (0.2) calculate polar moment of inertia
%-----------------------------------------
     % coefficients of the rectangle torsion bar for torsion deformation
 rt=[1.0, 1.2, 1.5, 2.0, 2.5, 3.0, 4.0, 5.0, 6.0, 8.0, 10.0, 50.0];
 bt=[0.141, 0.166, 0.196, 0.229, 0.249, 0.263, 0.281, 0.291, 0.299,
    0.307, 0.313, 0.333];

if Opt_section==1

  r0=h/b;
  If r0<1.0
   r0=1/r0; hx=h; h=b; b=hx;
  end

  ni=length(rt);
  b0=bt(1);

  if r0>=rt(end)
   b0=0.333;
  else
   for ii=1:ni-1          % linear interpolation for torsion coefficient
    if r0>=rt(ii)
     b0=bt(ii)+(bt(ii+1)-bt(ii))*(r0-rt(ii))/(rt(ii+1)-rt(ii));
    end
   end
  end

end

if polrmoment==0
  Jx=0;
elseif polrmoment==1
  if Opt_section==1; Jx=b0*h*b^3;
  elseif Opt_section==2; Jx=pi*(D^4-d^4)/32; end
else
  Jx=prop(9);
end
%--------------------------------
%  (0.3) calculate the shear modulus
%--------------------------------
 if shmodule==0
  G=0;
```

```
 elseif shmodule==1
   G=E/(2*(1+u));
 else
   G=prop(10);
 end
%-------------------------------------------------------------------------
%  (1) coordinate system transform matrix
%-------------------------------------------------------------------------
 cc=cos(al); ss=sin(al);
 c1=xi(1); c2=xi(2); c3=xi(3);
 la=sqrt(c1^2+c2^2);
if la==0
  tt=[ 0,  0,  1; -ss,  cc,  0; -cc, -ss,  0];
else
  d1=-c2/la; d2=c1/la; d3=0;
  e1=-c1*c3/la; e2=-c2*c3/la; e3=la;
  t1=[ 1,   0,   0; 0,  cc,  ss; 0, -ss,  cc];
  t2=[c1, c2, c3; d1, d2, d3; e1, e2, e3];
  tt=t1*t2;
end
 t0=zeros(3,3);
 T=[tt t0 t0 t0;
    t0 tt t0 t0.
    t0 t0 tt t0;
    t0 t0 t0 tt];
%-------------------------------------------------------------------------
%  (2) stiffness matrix
%-------------------------------------------------------------------------
%---------------------------------------------
%  (2.1) element matrix in local coordinate system
%---------------------------------------------
 ka=E*A/leng; kb=G*Jx/leng;
 kc=E*Iz/leng^3; kd=E*Iy/leng^3;
 k11=[ ka          0          0          0           0           0;
       0      12*kc          0          0           0      6*kc*leng;
       0          0      12*kd          0   -6*kd*leng           0;
       0          0          0         kb           0           0;
       0          0   -6*kd*leng        0   4*kd*leng^2          0;
       0    6*kc*leng        0          0           0    4*kc*leng^2];
 k12=[ -ka         0          0          0           0           0;
       0     -12*kc          0          0           0      6*kc*leng;
       0          0     -12*kd          0   -6*kd*leng           0;
       0          0          0        -kb           0           0;
       0          0    6*kd*leng        0   2*kd*leng^2          0;
       0   -6*kc*leng        0          0           0    2*kc*leng^2];
 k21=k12';
 k22=[ ka          0          0          0           0           0;
```

```
        0      12*kc          0         0          0     -6*kc*leng;
        0          0      12*kd         0    6*kd*leng             0;
        0          0          0        kb          0             0;
        0          0  6*kd*leng         0  4*kd*leng^2            0;
        0  -6*kc*leng         0         0          0   4*kc*leng^2];
 k0=[k11, k12; k21, k22];
%---------------------------------------------
% (2.2) element matrix in global coordinate system
%---------------------------------------------
 k=T'*k0*T;
%------------------------------------------------------------------------
% (3) mass matrix
%------------------------------------------------------------------------
if Opt_mass==1
%------------------------------
% (3.1) consistent mass matrix
%------------------------------
 ma=rho*A*leng/420; mb=70*Jx/A;
 m11=[140       0          0         0          0           0;
        0     156          0         0          0    22*leng;
        0       0        156         0   -22*leng           0;
        0       0          0      2*mb          0           0;
        0       0   -22*leng         0   4*leng^2           0;
        0  22*leng         0         0          0   4*leng^2];
 m12=[ 70       0          0         0          0           0;
        0      54          0         0          0   -13*leng;
        0       0         54         0    13*leng           0;
        0       0          0        mb          0           0;
        0       0   -13*leng         0  -3*leng^2           0;
        0  13*leng         0         0          0  -3*leng^2];
 m21=m12';
 m22=[140       0          0         0          0           0;
        0     156          0         0          0   -22*leng;
        0       0        156         0    22*leng           0;
        0       0          0      2*mb          0           0;
        0       0    22*leng         0   4*leng^2           0;
        0 -22*leng         0         0          0   4*leng^2];
 m0=ma*[m11, m12; m21, m22];

elseif Opt_mass==2
%--------------------------
% (3.2) lumped mass matrix
%--------------------------
 mass=rho*A*leng/2;
 mb=rho*Jx*leng/2;
 m0=mass*diag([1, 1, 1, mb/mass, 0, 0,...
           1, 1, 1, mb/mass, 0, 0]);
```

```
end
%-------------------------------------------
%  (3.3) element matrix in global coordinate system
%-------------------------------------------
 m=T'*m0*T;
%--------------------------------------------------------------------
%    The end
%--------------------------------------------------------------------
```

13. FrameElement32 函数

FrameElement32 函数的代码如下.

```
%--------------------------------------------------------------------
function [k,m]=FrameElement32(prop,leng,xi,al,Opt_section,Opt_mass)
%--------------------------------------------------------------------
%  Purpose:
%     To calculate stiffness and mass matrices for the 3-d frame element
%     (Timoshenko beam)
%     nodal dof: {u1  v1  w1  θx1  θy1  θz1  u2  v2  w2  θx2  θy2  θz2}
%  Synopsis:
%     [k,m]=FrameElement32(prop,leng,xi,al,Opt_section,Opt_mass)
%  Variable Description:
%     k, m - element stiffness matrix and element mass matrix
%     prop - the properties of materials and geometry:
%     leng - element length
%     xi - the first row of the coordinate transform matrix between the
%     local and global axes
%       xi(1)=cos(x,x'), xi(2)=cos(x,y'), xi(3)=cos(x,z')
%     al - angle between the reference coordinate system and the local
%     coordinate system
%       for the space element
%     Opt_section - option for type of cross-section
%       = 1 - rectangular cross-section
%       = 2 - circular cross-section
%     Opt_mass - option for mass matrix
%             = 1 - consistent mass matrix
%             = 2 - lumped mass matrix
%--------------------------------------------------------------------
% (0) calculation of the constants
%--------------------------------------------------------------------
%-------------------------------
%  (0.1) evaluation of the constants
%-------------------------------
E=prop(1);                                % elastic modulus
u=prop(2);                                 % Poisson's ratio
rho=prop(3);                % mass density (mass per unit volume)
if Opt_section==1
```

```
  h=prop(4);                   % height of beam cross-section
  b=prop(5);                   % width of beam cross-section
elseif Opt_section==2
  D=prop(4);               % outer diameter of beam cross-section
  d=prop(5);               % inner diameter of beam cross-section
end
A=prop(6);             % area of beam cross-section
Iy=prop(7);        % 2nd moment of inertia of cross-section about axis y
Iz=prop(8);        % 2nd moment of inertia of cross-section about axis z

polrmoment=prop(9);        % selection of the polar moment of inertia
shmodule=prop(10);         % selection of shear modulus
%------------------------------------
%  (0.2) calculate polar moment of inertia
%------------------------------------
     % coefficients of the rectangle torsion bar for torsion deformation
 rt=[1.0, 1.2, 1.5, 2.0, 2.5, 3.0, 4.0, 5.0, 6.0, 8.0, 10.0, 50.0];
 bt=[0.141, 0.166, 0.196, 0.229, 0.249, 0.263, 0.281, 0.291, 0.299,
    0.307, 0.313, 0.333];

if Opt_section==1

  r0=h/b;
  if r0<1.0
    r0=1/r0; hx=h; h=b; b=hx;
  end

  ni=length(rt);
  b0=bt(1);

  if r0>=rt(end)
    b0=0.333;
  else
    for ii=1:ni-1         % linear interpolation for torsion coefficient
      if r0>=rt(ii)
        b0=bt(ii)+(bt(ii+1)-bt(ii))*(r0-rt(ii))/(rt(ii+1)-rt(ii));
      end
    end
  end

end


if polrmoment==0
  Jx=0;
elseif polrmoment==1
  if Opt_section==1; Jx=b0*h*b^3;
```

```
  elseif Opt_section==2; Jx=pi*(D^4-d^4)/32; end
else
  Jx=prop(9);
end
%------------------------------
%  (0.3) calculate the shear modulus
%------------------------------
 if shmodule==0
   G=0;
 elseif shmodule==1
   G=E/(2*(1+u));
 else
   G=prop(10);
 end
%----------------------------------------------
%  (0.4) selection of correction factor for shear energy
%----------------------------------------------
if Opt_section==1
%---------------------------------------------------------------------------
   ck=6/5;          % correction factor of the rectangular cross-section
elseif Opt_section==2
   ck=10/9;         % correction factor of the circular cross-section
end
%---------------------------------------------------------------------------
%  (1) coordinate system transform matrix
%---------------------------------------------------------------------------
 cc=cos(al); ss=sin(al);
 c1=xi(1);   c2=xi(2);   c3=xi(3);
 la=sqrt(c1^2+c2^2);

if la==0
  tt=[ 0,  0,  1; -ss,  cc,  0; -cc, -ss,  0];
else
  d1=-c2/la; d2=c1/la; d3=0;
  e1=-c1*c3/la; e2=-c2*c3/la; e3=la;

  t1=[ 1,   0,   0; 0,  cc,  ss; 0, -ss,  cc];
  t2=[c1, c2, c3; d1, d2, d3; e1, e2, e3];

  tt=t1*t2;
end

 t0=zeros(3,3);
 T=[tt t0 t0 t0;...
   t0 tt t0 t0;...
   t0 t0 tt t0;...
   t0 t0 t0 tt];
```

```
%---------------------------------------------------------------------------
% (2) element matrix in local coordinate system
%---------------------------------------------------------------------------
%---------------------
% (2.1) stiffness matrix
%---------------------
 ka=E*A/leng;   kb=G*Jx/leng;
 kc=E*Iz/leng;  kd=E*Iy/leng;
 ke=G*A/(4*ck*leng);

 k11=[ ka        0          0       0          0            0;
       0       4*ke         0       0          0       2*ke*leng;
       0        0         4*ke      0     -2*ke*leng         0;
       0        0          0       kb          0             0;
       0        0    -2*ke*leng     0    ke*leng^2+kd          0;
       0   2*ke*leng        0       0          0    ke*leng^2+kc];
 k12=[-ka        0          0       0          0            0;
       0      -4*ke         0       0          0       2*ke*leng;
       0        0        -4*ke      0     -2*ke*leng         0;
       0        0          0      -kb          0             0;
       0        0     2*ke*leng     0    ke*leng^2-kd          0;
       0  -2*ke*leng        0       0          0    ke*leng^2-kc];
 k21=k12';
 k22=[ ka        0          0       0          0            0;
       0       4*ke         0       0          0      -2*ke*leng;
       0        0         4*ke      0      2*ke*leng         0;
       0        0          0       kb          0             0;
       0        0     2*ke*leng     0    ke*leng^2+kd          0;
       0  -2*ke*leng        0       0          0    ke*leng^2+kc];
 k0=[k11, k12; k21, k22];

 if Jx==0
   k0(:,3)=0; k0(:,5)=0; k0(:,9)=0; k0(:,11)=0;
 end
%----------------------------------------
% (2.2) element matrix in global coordinate system
%----------------------------------------
 k=T'*k0*T;
%---------------------------------------------------------------------------
% (3) mass matrix
%---------------------------------------------------------------------------

if Opt mass==1
%----------------------------
% (3.1) consistent mass matrix
%----------------------------
 ma=rho*A*leng/6; mb=rho*Jx*leng/6;
```

```
  m11=[ 2         0         0         0         0         0;
        0         2         0         0         0         0;
        0         0         2         0         0         0;
        0         0         0     2*mb/ma       0         0;
        0         0         0         0         0         0;
        0         0         0         0         0         0];
  m12=[ 1         0         0         0         0         0;
        0         1         0         0         0         0;
        0         0         1         0         0         0;
        0         0         0       mb/ma       0         0;
        0         0         0         0         0         0;
        0         0         0         0         0         0];
  m21=m12';
  m22=[ 2         0         0         0         0         0;
        0         2         0         0         0         0;
        0         0         2         0         0         0;
        0         0         0     2*mb/ma       0         0;
        0         0         0         0         0         0;
        0         0         0         0         0         0];
  m0=ma*[m11, m12; m21, m22];

else
%------------------------
%  (3.2) lumped mass matrix
%------------------------
  mass=rho*A*leng/2;
  mb=rho*Jx*leng/2;
  m0=mass*diag([1, 1, 1, mb/mass, 0, 0,...
           1, 1, 1, mb/mass, 0, 0]);

end
%-------------------------------------------
%  (3.3) element matrix in global coordinate system
%-------------------------------------------
 m=T'*m0*T;
%-------------------------------------------------------------------------
%    The end
%-------------------------------------------------------------------------
```

14. ImpulseRespt 函数

ImpulseRespt 函数的代码如下.

```
%-------------------------------------------------------------------------
function [eta,y,omega1,sdof2]=ImpulseRespt(kk,mm,fd,u,t,C,q0,dq0,a,b)
%-------------------------------------------------------------------------
%  Purpose:
```

```
%    The function subroutine ImpulseRespt.m calculates impulse response
%    of a damping or undamping structural system using modal analysis.
%    It uses modal coordinate equations to compute the modal responses
%    analytically, then convert the modal responses into physical
%    responses through the coordinate transformation.
%  Synopsis:
%    [eta,y,omega1,sdof2]=ImpulseRespt(kk,mm,fd,u,t,C,q0,dq0,a,b)
%  Variable Description:
%    Input parameters:
%      kk, mm - system stiffness and mass matrices
%      fd - input or forcing influence matrix
%      u - number of the loads
%      t - time of evaluation
%      C - output matrix
%      q0, dq0 - initial conditions
%      a, b - parameters for proportional damping [C]=a[M]+b[K]
%    Output parameters:
%      eta - modal coordinate response
%      y - physical coordinate response
%      omega - natural frequency
%-----------------------------------------------------------------------
%  (1) Solve the eigenvalue problem
%-----------------------------------------------------------------------
t=t';
[sdof,n1]=size(kk);
[nstep,n2]=size(t);

[V,D]=eig(kk,mm);                % compute the eigenvalues and eigenvectors
[lambda,ki]=sort(diag(D));        % sort the eigenvalues and eigenvectors
omega=sqrt(lambda);                          % natural frequencies
omega1=sqrt(lambda)/(2*pi);                  % the frequency vector in Hz

V=V(:,ki);
%-----------------------------------------------------------------------
%  (2) Check the eigenvalues
%-----------------------------------------------------------------------
                % check whether the eigenvalues are infinite and eliminate
                  % the eigenvectors associated with the bad eigenvalues.
jk=0;

for ii=1:sdof                  % loop for find the infinite in omega
  check=omega(ii);
  if check>1.0e12
    jk=jk+1;                   % location of the infinite frequency
    omi(jk)=ii;             % storing the location of the infinite frequency
  end
end
```

```
sdof2=sdof-jk;
V1=[V(:,1:sdof2)];                     % truncate the modal vectors
%----------------------------------------------------------------------
%  (3) Normalize the eigenvectors and compute parameters
%----------------------------------------------------------------------
Factor=diag(V1'*mm*V1);
Vnorm=V1*inv(sqrt(diag(Factor)));     %  eigenvectors are normalized
omega2=diag(sqrt(Vnorm'*kk*Vnorm));         % natural frequencies

Fnorm=Vnorm'*fd;                       % modal input force vector
%----------------------------------------------------------------------
%  (4) Compute modal damping matrix from the proportional damping matrix
%----------------------------------------------------------------------
Modamp=Vnorm'*(a*mm+b*kk)*Vnorm;       % form the Rayleigh damping
zeta=diag((1/2)*Modamp*inv(diag(omega2)));        % the damping ratio

if (max(zeta) >= 1),
  disp('Warning - Your maximum damping ratio is grater than or equal to 1')
  disp('You have to reselect a and b ')
  pause
  disp('If you want to continue, type return key')
end
%----------------------------------------------------------------------
%  (5) Find out impulse response of each modal coordinate analytically
%----------------------------------------------------------------------
eta0=Vnorm'*mm*q0; deta0=Vnorm'*mm*dq0;
   % initial conditions for modal coordinates both displacement and velocity
eta=zeros(nstep,sdof2);

 for i=1:sdof2              % responses are calculated for n modes
   omegad=omega(i)*sqrt(1-zeta(i)^2);
   phase=omegad*t;
   Exx=exp(-zeta(i)*omega(i)*t);

   C1=eta0(i);
   C2=(deta0(i)+eta0(i)*zeta(i)*omega(i))/omegad;

   XX=Fnorm(i,u)/omegad;

   eta(:,i)=C1*Exx.*cos(phase)+C2*Exx.*sin(phase)...
           +XX*Exx.*sin(phase);
                             % response for the impulse excitation
 end
%----------------------------------------------------------------------
%  (6) Convert modal coordinate responses to physical coordinate responses
%----------------------------------------------------------------------
```

```
eta=eta';
y=C*Vnorm*eta;

if (a+b)==0
  disp('The response results of undamping system')
else
  disp('The response results of damping system')
end

disp('The excitation is impulse force')
%-----------------------------------------------------------------------
%    The end
%-----------------------------------------------------------------------
```

15. StepRespt 函数

StepRespt 函数的代码如下.

```
%-----------------------------------------------------------------------
function [eta,y,omega1,sdof2]=StepRespt(kk,mm,fd,u,t,C,q0,dq0,a,b)
%-----------------------------------------------------------------------
% Purpose:
%   The function subroutine StepRespt.m calculates step response for a
%   damping or undamping structural system using modal analysis. It
%   uses modal coordinate equations to compute the modal responses
%   analytically, then convert the modal responses into physical
%   responses through the coordinate transformation.
% Synopsis:
%   [eta,y,omega1,sdof2]=StepRespt(kk,mm,fd,u,t,C,q0,dq0,a,b)
% Variable Description:
%   Input parameters:
%     kk, mm - System stiffness and mass matrices
%     fd - Input or forcing influence matrix
%     u - number of the loads
%     t - Time of evaluation
%     C - Output matrix
%     q0, dq0 - Initial conditions
%     a, b - Parameters for proportional damping [C]=a[M]+b[K]
%   Outpur parameters:
%     eta - modal coordinate response
%     y - physical coordinate response
%     omega - natrural frequency
%-----------------------------------------------------------------------
% (1) Solve the eigenvalue problem
%-----------------------------------------------------------------------
t=t';
[sdof,n1]=size(kk);
[nstep,n2]=size(t);
```

```
[V,D]=eig(kk,mm);            % compute the eigenvalues and eigenvectors
[lambda,ki]=sort(diag(D));       % sort the eigenvalues and eigenvectors
omega=sqrt(lambda);                   % natural frequencies
omega1=sqrt(lambda)/(2*pi);           % the frequency vector in Hz

V=V(:,ki);
%-------------------------------------------------------------------------
%  (2) Check the eigenvalues
%-------------------------------------------------------------------------
               % check whether the eigenvalues are infinite and eliminate
               % the eigenvectors associated with the bad eigenvalues
jk=0;

for ii=1:sdof                    % loop for find the infinite in omega
  check=omega(ii);
  if check>1.0e12
    jk=jk+1;              % location of the infinite frequency
    omi(jk)=ii;           % storing the location of the infinite frequency
  end
end

sdof2=sdof-jk;
V1=[V(:,1:sdof2)];                      % truncate the modal vectors
%-------------------------------------------------------------------------
%  (3) Normalize the eigenvectors and compute parameters
%-------------------------------------------------------------------------
Factor=diag(V1'*mm*V1);
Vnorm=V1*inv(sqrt(diag(Factor)));        %  eigenvectors are normalized
omega2=diag(sqrt(Vnorm'*kk*Vnorm));      % natural frequencies

Fnorm=Vnorm'*fd;                      % modal input force vector
%-------------------------------------------------------------------------
%  (4) Compute modal damping matrix from the proportional damping matrix
%-------------------------------------------------------------------------
Modamp=Vnorm'*(a*mm+b*kk)*Vnorm;      % form the Rayleigh damping
zeta=diag((1/2)*Modamp*inv(diag(omega2)));          % the damping ratio

if (max(zeta) >= 1),
  disp('Warning - Your maximum damping ratio is grater than or equal to 1')
  disp('You have to reselect a and b ')
  pause
  disp('If you want to continue, type return key')
end
%-------------------------------------------------------------------------
%  (5) Find out step response of each modal coordinate analytically
%-------------------------------------------------------------------------
```

```
eta0=Vnorm'*mm*q0; deta0=Vnorm'*mm*dq0;
    % initial conditions for modal coordinates both displacement and velocity
eta=zeros(nstep,sdof2);

  for i=1:sdof2                  % responses are calculated for n modes
    omegad=omega(i)*sqrt(1-zeta(i)^2);
    phase=omegad*t;
    Exx=exp(-zeta(i)*omega(i)*t);

    C1=eta0(i);
    C2=(deta0(i)+eta0(i)*zeta(i)*omega(i))/omegad;

    D1=zeta(i)*omega(i)/omegad;
    II=ones(nstep,1);

    XX=Fnorm(i,u)/(omegad^2+zeta(i)^2*omega(i)^2);

    eta(:,i)=C1*Exx.*cos(phase)+C2*Exx.*sin(phase)...
            + XX*(II-Exx.*cos(phase)-D1*Exx.*sin(phase));
                              % response for the step excitation
  end
%--------------------------------------------------------------------------
%  (6) Convert modal coordinate responses to physical coordinate responses
%--------------------------------------------------------------------------
eta=eta';
y=C*Vnorm*eta;

if (a+b)==0
  disp('The response results of undamping system')
else
  disp('The response results of damping system')
end

disp('The excitation is step force')
%--------------------------------------------------------------------------
%    The end
%--------------------------------------------------------------------------
```

16. HarmonicRespt 函数

HarmonicRespt 函数的代码如下.

```
%--------------------------------------------------------------------------
function [eta,y,omega1,sdof2]=HarmonicRespt(kk,mm,fd,omega0,t,C,q0,dq0,a,b)
%--------------------------------------------------------------------------
% Purpose:
%   The function subroutine HarmonicRespt.m calculates harmonic response
%   for a damping or undamping structural system using modal analysis.
```

```
%   It uses modal coordinate equations to compute modal responses analytically,
%   then convert the modal responses into physical responses through
%   the coordinate transformation.
% Synopsis:
%    [eta,y,omega1,sdof2]= HarmonicRespt (kk,mm,fd,omega0,t,C,q0,dq0,a,b)
% Variable Description:
%   Input parameters:
%     kk, mm - System stiffness and mass matrices
%     fd - Input or forcing influence matrix
%     omega0 - Frequency of the excitation
%     t - Time of evaluation
%     C - Output matrix
%     q0, dq0 - Initial conditions
%     a, b - Parameters for proportional damping [C]=a[M]+b[K]
%   Output parameters:
%     eta - modal coordinate response
%     y - physical coordinate response
%     omega1 - natural frequency
%-----------------------------------------------------------------------
%  (1) Solve the eigenvalue problem
%-----------------------------------------------------------------------
t=t';
[sdof,n1]=size(kk);
[nstep,n2]=size(t);

[V,D]=eig(kk,mm);              % compute the eigenvalues and eigenvectors
[lambda,ki]=sort(diag(D));        % sort the eigenvalues and eigenvectors
omega=sqrt(lambda);               % natural frequencies
omega1=sqrt(lambda)/(2*pi);           % the frequency vector in Hz

V=V(:,ki);
%-----------------------------------------------------------------------
%  (2) Check the eigenvalues
%-----------------------------------------------------------------------
              % check whether the eigenvalues are infinite and eliminate
              % the eigenvectors associated with the bad eigenvalues
jk=0;

for ii=1:sdof                    % loop for find the infinite in omega

 check=omega(ii);
 if check>1.0e12
   jk=jk+1;                   % location of the infinite frequency
   omi(jk)=ii;           % storing the location of the infinite frequency
 end

end
```

```
sdof2=sdof-jk;
V1=[V(:,1:sdof2)];                    % truncate the modal vectors
%----------------------------------------------------------------------
%  (3) Normalize the eigenvectors and compute parameters
%----------------------------------------------------------------------
Factor=diag(V1'*mm*V1);
Vnorm=V1*inv(sqrt(diag(Factor)));       %  eigenvectors are normalized
omega2=diag(sqrt(Vnorm'*kk*Vnorm));          % natural frequencies

Fnorm=Vnorm'*fd;                        % modal input force vector
%----------------------------------------------------------------------
%  (2) Compute modal damping matrix from the proportional damping matrix
%----------------------------------------------------------------------
Modamp=Vnorm'*(a*mm+b*kk)*Vnorm;        % form the Rayleigh damping
zeta=diag((1/2)*Modamp*inv(diag(omega2)));      % the damping ratio

if (max(zeta) >= 1),
  disp('Warning - Your maximum damping ratio is grater than or equal to 1')
  disp('You have to reselect a and b ')
  pause
  disp('If you want to continue, type return key')
end
%----------------------------------------------------------------------
%  (3) Find out harmonic response of each modal coordinate analytically
%----------------------------------------------------------------------
eta0=Vnorm'*mm*q0; deta0=Vnorm'*mm*dq0;
    % initial conditions for modal coordinates both displacement and velocity
eta=zeros(nstep,sdof2);

phase0=omega0*t;

for i=1:sdof2                     % responses are obtained for n modes
  gama=omega0/omega(i);
  omegad=omega(i)*sqrt(1-zeta(i)^2);
  phase=omegad*t;
  Exx=exp(-zeta(i)*omega(i)*t);

  C1=eta0(i);
  C2=(deta0(i)+eta0(i)*zeta(i)*omega(i))/omegad;

  X0=sqrt((1-gama^2)^2+(2*zeta(i)*gama)^2);
  XX=Fnorm(i)/(omega(i)^2*X0);
  XP=atan((2*zeta(i)*gama)/(1-gama^2));

  D1=(zeta(i)*omega(i)*cos(XP)+omega0*sin(XP))/omegad;
  D2=cos(XP);
```

```
  eta(:,i)=C1*Exx.*cos(phase)+C2*Exx.*sin(phase)...
          -XX*Exx.*(D1*sin(phase)+D2*cos(phase))...
          +XX*cos(phase0-XP);
          % response included transient-state for the harmonic excitation
end
%------------------------------------------------------------------------
% (4) Convert modal coordinate responses to physical coordinate responses
%------------------------------------------------------------------------
eta=eta';
y=C*Vnorm*eta;

if (a+b)==0
  disp('The response results of undamping system')
else
  disp('The response results of damping system')
end

disp('The excitation is harmonic force')
%------------------------------------------------------------------------
%    The end
%------------------------------------------------------------------------
```

17. femFFT 函数

femFFT 函数的代码如下.

```
%------------------------------------------------------------------------
function [yfft,freq]=femFFT(y,t)
%------------------------------------------------------------------------
% Purpose:
%    This function subroutine calculates Fast Fourier Transform (FFT) of
%    the time domain signal. The time domain data are provided with
%    corresponding time interval.
% Synopsis:
%    [yfft, freq]=femFFT(y,t)
% Variable Description:
%    Input parameters - y : Time domain data n by 1
%                       t : Time interval for y  of n by 1 size
%    Output - yfft : Absolute value of FFT of the time domain data y
%             freq : Frequency axis values
% Notes:
%   The number of data points for y should be power of 2, and
%   truncation is needed to achieve the requirement
%------------------------------------------------------------------------
% (1) Compute number of data points and sampling time interval
%------------------------------------------------------------------------
ntime=max(size(t));
```

```
dt=(t(1,ntime)-t(1,1))/(ntime-1);
%-----------------------------------------------------------------------
%  (2) Truncate the data points of y
%-----------------------------------------------------------------------
     % Extract data points at the power of 2. Truncate extra data points
     % so that the final number of data points is in the power of two and
     % also as close as possible to the given number of data points
N=fix(log10(ntime)/log10(2));
%-----------------------------------------------------------------------
%  (3) Calculate FFT
%-----------------------------------------------------------------------
                          %  Calculate FFT of the time domain data and
                          %  take absolute values of the result
yfft=fft(y(1:2^N,:));
yfft=abs(yfft(1:2^N/2,:))*dt;
%-----------------------------------------------------------------------
%  (4) Calculate frequency vector
%-----------------------------------------------------------------------
        %  Set up the frequency scale from the given sampling interval.
        %  Apply the Nyquist criterion to establish the maximum frequency
freq0=0;
freqf= (1/dt)/2;                 %  Maximum or final frequency value
df=freqf/(2^N/2);                         %  Frequency interval

freq=0:df:freqf-df;                       %  Frequency axis values
%-----------------------------------------------------------------------
%    The end
%-----------------------------------------------------------------------
```

第6章 弹性问题

6.1 平面问题

弹性力学平面问题可分为平面应力问题和平面应变问题. 对于线弹性各向同性体，分析时除弹性矩阵不同外，其他都相同. 因此下面在分析时，除在给出与弹性矩阵有关的显式时加以说明外，其他地方不加以区别.

对于平面应力问题，弹性矩阵 $\boldsymbol{D}$ 为

$$\boldsymbol{D}=\frac{E}{1-\mu^2}\begin{bmatrix}1 & \mu & 0\\ \mu & 1 & 0\\ 0 & 0 & \dfrac{1-\mu}{2}\end{bmatrix} \tag{6-1}$$

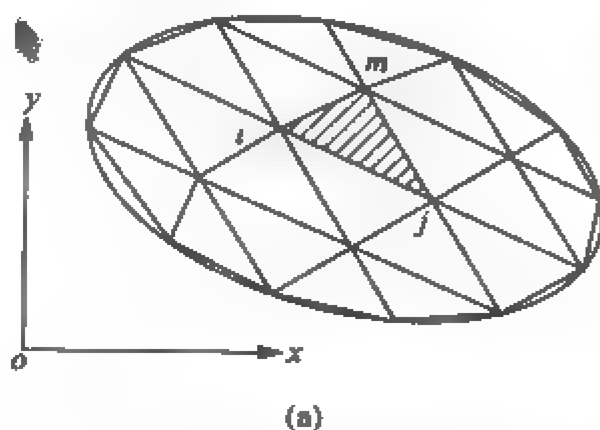
对于平面应变问题，弹性矩阵 $\boldsymbol{D}$ 为

$$\boldsymbol{D}=\frac{E(1-\mu)}{(1+\mu)(1-2\mu)}\begin{bmatrix}1 & \dfrac{\mu}{1-\mu} & 0\\ \dfrac{\mu}{1-\mu} & 1 & 0\\ 0 & 0 & \dfrac{1-2\mu}{2(1-\mu)}\end{bmatrix} \tag{6-2}$$

6.1.1 常应变三角形单元

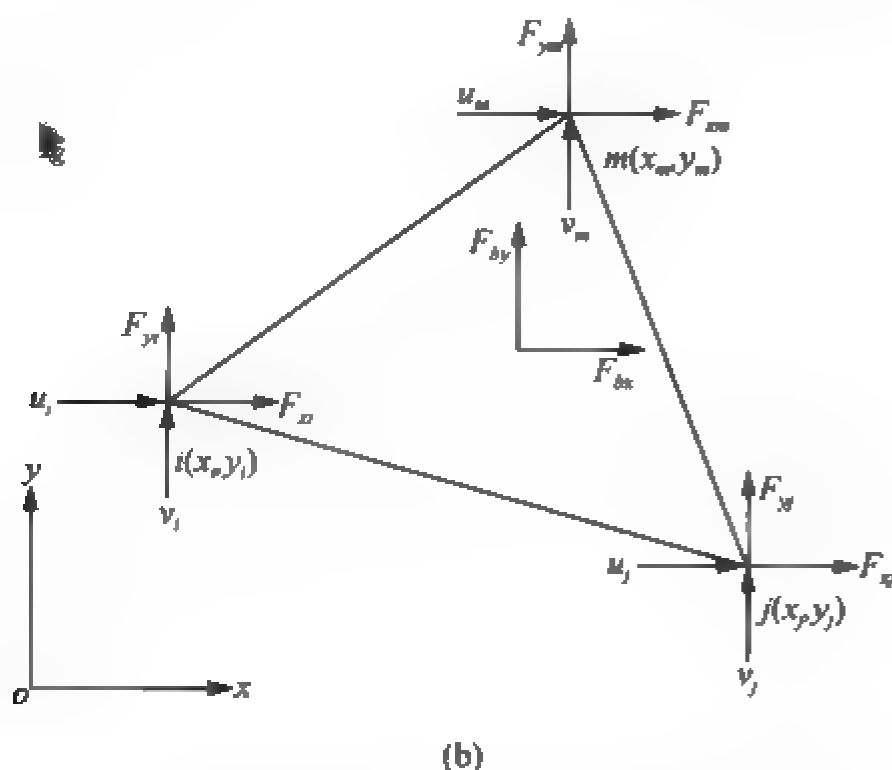
1. 离散化

有限元法应用于结构分析时，第一步就是把结构离散化. 对于平面问题，最简单、最常用的离散方式是将其分割为有限个三角形单元，单元之间在三角形顶点上相连，如图 6.1(a) 所示. 这种单元虽然简单，但也可用来拟合复杂边界体形. 虽然当边界为曲线时，存在以三角形直边来代替而带来离散误差，但比利用矩形单元进行离散时的离散误差小得多.



(a)

图 6.1 常应变三角形单元示意图



(b)

图6.1 (续)

图6.1(b)是从如图6.1(a)所示离散体系中取出的第e个单元，其节点i、j和m按逆时针排列. 每个节点位移在单元平面内有两个分量. 整个单元将有6个节点位移分量，可用列阵表示为

$$\boldsymbol{\delta}^e=\begin{bmatrix}\boldsymbol{\delta}_i^{\mathrm{T}} & \boldsymbol{\delta}_j^{\mathrm{T}} & \boldsymbol{\delta}_m^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}=\begin{bmatrix}u_i & v_i & u_j & v_j & u_m & v_m\end{bmatrix}^{\mathrm{T}} \tag{6-3}$$

同样，单元节点力列阵为

$$\boldsymbol{F}^e=\begin{bmatrix}\boldsymbol{F}_i^{\mathrm{T}} & \boldsymbol{F}_j^{\mathrm{T}} & \boldsymbol{F}_m^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}=\begin{bmatrix}F_{xi} & F_{yi} & F_{xj} & F_{yj} & F_{xm} & F_{ym}\end{bmatrix}^{\mathrm{T}} \tag{6-4}$$

单元上作用的体积力记为

$$\boldsymbol{F}_b^e=\begin{Bmatrix}F_{bx}\\ F_{by}\end{Bmatrix} \tag{6-5}$$

若单元的边界是物理边界，并且该边界有表面力作用，该表面力记为

$$\boldsymbol{F}_s^e=\begin{Bmatrix}F_{sx}\\ F_{sy}\end{Bmatrix} \tag{6-6}$$

体积力和表面力表达式中的矩阵元素均是沿坐标方向的分布载荷集度.

2. 用面积坐标建立单元位移场

对于三角形单元，完全可以同杆系单元一样，在直角坐标系下采用广义坐标法建立形函数及单元位移场. 但下面将引入的面积坐标除了也可以确定点的位置外，对三角形单元的分析还具有许多优越性.

1) 面积坐标的定义

设p为三角形单元中的一点，与3个顶点i, j, m相连，则可将$\triangle ijm$分割成3小块，如图6.2所示，分别记这3个小三角形面积为

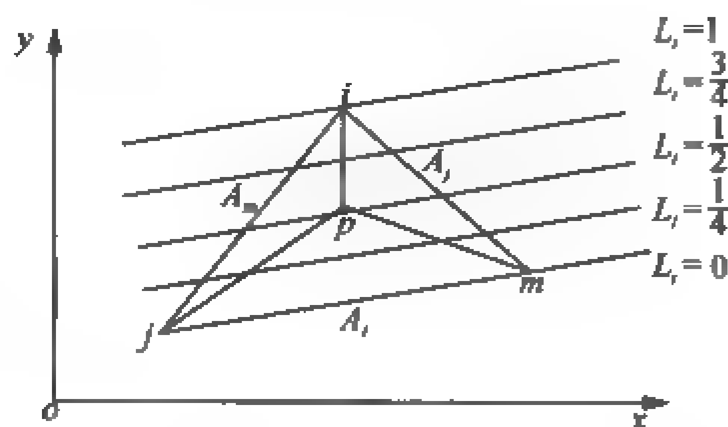


图 6.2　面积坐标示意图

$$A_i = A_{pjm}, \quad A_j = A_{pmi}, \quad A_k = A_{pij} \tag{6-7}$$

则显然存在如下恒等关系

$$A = A_i + A_j + A_m = A_{ijm} \tag{6-8}$$

p 点位置可由 3 个比值来确定

$$p(L_i, L_j, L_m)$$

其中

$$L_i = A_i/A, \quad L_j = A_j/A, \quad L_m = A_m/A \tag{6-9}$$

称 L_i, L_j, L_m 为面积坐标.

面积坐标的特点如下.

- 三角形内与节点 i 的对边 j-m 平行的直线上的诸点有相同的 L_i.
- 三角形 3 个角点的面积坐标是 $i(1,0,0)$，$j(0,1,0)$，$m(0,0,1)$.
- 三角形 3 条边的边方程是
  j-m 边　$L_i = 0$；　m-i 边　$L_j = 0$；　i-j 边　$L_m = 0$
- 3 个面积坐标并不相互独立，3 个面积坐标间必然满足

$$L_i + L_j + L_k = 1 \tag{6-10}$$

2) 面积坐标与直角坐标之间的关系

设三角形的 3 个顶点在直角坐标系中的位置是 $i(x_i, y_i), j(x_j, y_j), m(x_m, y_m)$，$p$ 点坐标是 (x, y)，如图 6.3 所示. 将 A, A_i, A_j, A_m 等用直角坐标表示，就可以建立面积坐标和直角坐标的转换关系.

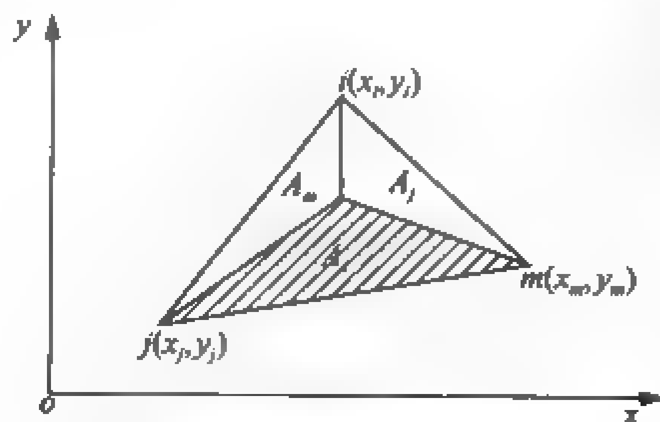


图 6.3　面积坐标与直角坐标的关系

$$A_i=\frac{1}{2}\begin{vmatrix}1 & x & y\\ 1 & x_j & y_j\\ 1 & x_m & y_m\end{vmatrix}$$

$$=\frac{1}{2}\left[\left(x_j y_m - y_j x_m\right)+\left(y_j - y_m\right)x+\left(x_m - x_j\right)y\right]$$

$$=\frac{1}{2}(a_i+b_i x+c_i y)$$

角坐标轮换规则为$i\to j\to m\to i$

$$L_i=\frac{A_i}{A}=\frac{1}{2A}(a_i+b_i x+c_i y)\qquad (i\to j\to m\to i)$$

式中，a_i、b_i、c_i分别为

$$a_i=x_j y_m - y_j x_m$$

$$b_i=y_j-y_m\qquad (i\to j\to m\to i)$$

$$c_i=x_m-x_j$$

面积坐标用直角坐标表示的矩阵表达式为

$$\begin{Bmatrix}L_i\\ L_j\\ L_m\end{Bmatrix}=\frac{1}{2A}\begin{bmatrix}a_i & b_i & c_i\\ a_j & b_j & c_j\\ a_m & b_m & c_m\end{bmatrix}\begin{Bmatrix}1\\ x\\ y\end{Bmatrix}\tag{6-11}$$

将L_i,L_j,L_m分别乘以x_i,x_j,x_m，然后相加，可以得到

$$x=x_i L_i+x_j L_j+x_m L_m\tag{6-12}$$

同理

$$y=y_i L_i+y_j L_j+y_m L_m\tag{6-13}$$

式(6-12)和式(6-13)与式(6-10)可合并表示为

$$\begin{Bmatrix}1\\ x\\ y\end{Bmatrix}=\begin{bmatrix}1 & 1 & 1\\ x_i & x_j & x_m\\ y_i & y_j & y_m\end{bmatrix}\begin{Bmatrix}L_i\\ L_j\\ L_m\end{Bmatrix}\tag{6-14}$$

建立了坐标间的变换关系，按求导法可得

$$\left.\begin{aligned}\frac{\partial}{\partial x}&=\frac{1}{2A}\left(b_i\frac{\partial}{\partial L_i}+b_j\frac{\partial}{\partial L_j}+b_m\frac{\partial}{\partial L_m}\right)\\ \frac{\partial}{\partial y}&=\frac{1}{2A}\left(c_i\frac{\partial}{\partial L_i}+c_j\frac{\partial}{\partial L_j}+c_m\frac{\partial}{\partial L_m}\right)\end{aligned}\right\}\tag{6-15}$$

利用数学知识可以证明

$$\int_v L_i^{\alpha}L_j^{\beta}L_m^{\gamma}\mathrm{d}V=\frac{\alpha!\ \beta!\ \gamma!}{(\alpha+\beta+\gamma+2)!}2hA\tag{6-16}$$

$$\int_{S_e} L_i^{\alpha}L_j^{\beta}\mathrm{d}s=\frac{\alpha!\beta!}{(\alpha+\beta+1)!}hl_{ij}\tag{6-17}$$

式中：α、β、γ为面积坐标的幂；l_{ij}为ij边的长度.

3) 常应变三角形单元的位移场

常应变三角形单元的形函数取面积坐标，即

$$N_i = L_i, \quad N_j = L_j, \quad N_m = L_m \tag{6-18}$$

由此可得形函数矩阵为

$$\boldsymbol{N} = \begin{bmatrix} N_i & 0 & N_j & 0 & N_m & 0 \\ 0 & N_i & 0 & N_j & 0 & N_m \end{bmatrix} = \begin{bmatrix} N_i \boldsymbol{I}_2 & N_j \boldsymbol{I}_2 & N_m \boldsymbol{I}_2 \end{bmatrix} \tag{6-19}$$

则单元内任意一点的位移可表示为

$$\boldsymbol{d} = \begin{Bmatrix} u \\ v \end{Bmatrix} = \boldsymbol{N} \boldsymbol{\delta}^e \tag{6-20}$$

3. 单元应变和单元应力

有了单元的位移模式，就可以利用平面问题的几何方程

$$\boldsymbol{\varepsilon} = \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} = \begin{Bmatrix} \partial u/\partial x \\ \partial v/\partial y \\ \partial u/\partial y + \partial v/\partial x \end{Bmatrix} \tag{6-21}$$

求得应变分量. 将式(6-20)代入式(6-21)得到

$$\boldsymbol{\varepsilon} = \frac{1}{2A} \begin{bmatrix} b_i & 0 & b_j & 0 & b_m & 0 \\ 0 & c_i & 0 & c_j & 0 & c_m \\ c_i & b_i & c_j & b_j & c_m & b_m \end{bmatrix} = \boldsymbol{B} \boldsymbol{\delta}^e \tag{6-22}$$

式中的单元应变矩阵 $\boldsymbol{B}$ 可以写成分块形式

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{B}_i & \boldsymbol{B}_j & \boldsymbol{B}_m \end{bmatrix} \tag{6-23}$$

而子矩阵

$$\boldsymbol{B}_i = \frac{1}{2A} \begin{bmatrix} b_i & 0 \\ 0 & c_i \\ c_i & b_i \end{bmatrix} \qquad (i,\ j,\ m) \tag{6-24}$$

根据式(6-22)可知，由于 A 和 b_i、b_j、b_m、c_i、c_j 和 c_m 等都是常数，故矩阵 $\boldsymbol{B}$ 中的元素都是常量，因而单元中任意一点的应变分量 ε_x、ε_y 和 γ_{xy} 也都是常量，故通常称这种单元为常应变单元.

将式(6-22)代入应力应变关系式，得到

$$\boldsymbol{\sigma} = \boldsymbol{D}\boldsymbol{B}\boldsymbol{\delta}^e = \boldsymbol{S}\boldsymbol{\delta}^e \tag{6-25}$$

式中，$\boldsymbol{S}$ 为应力矩阵，以分块形式表示为

$$\boldsymbol{S} = \boldsymbol{D}\begin{bmatrix} \boldsymbol{B}_i & \boldsymbol{B}_j & \boldsymbol{B}_m \end{bmatrix} = \begin{bmatrix} \boldsymbol{S}_i & \boldsymbol{S}_j & \boldsymbol{S}_m \end{bmatrix} \tag{6-26}$$

对于平面应力问题，的子矩阵可写为

$$\boldsymbol{S}_i = \frac{E}{2(1-\mu^2)A} \begin{bmatrix} b_i & \mu c_i \\ \mu b_i & c_i \\ \frac{1-\mu}{2} c_i & \frac{1-\mu}{2} b_i \end{bmatrix} \qquad (i,\ j,\ m) \tag{6-27}$$

对于平面应变问题，$\boldsymbol{S}$的子矩阵可写为

$$\boldsymbol{S}_i=\frac{E(1-\mu)}{2(1+\mu)(1-2\mu)A}\begin{bmatrix} b_i & \dfrac{\mu}{1-\mu}c_i \\ \dfrac{\mu}{1-\mu}b_i & c_i \\ \dfrac{1-2\mu}{2(1-\mu)}c_i & \dfrac{1-2\mu}{2(1-\mu)}b_i \end{bmatrix} \tag{6-28}$$

4. 单元刚度矩阵和等效节点力

1) 单元刚度矩阵

根据最小势能原理，可以推导得到三角形单元的单元刚度矩阵为

$$\boldsymbol{K}^e=\iiint_{\Omega}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{B}\mathrm{d}\Omega=\iint_{\Delta}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{B}h\mathrm{d}x\mathrm{d}y \tag{6-29}$$

式中，h是单元的厚度. 由于应变矩阵是常数矩阵，若单元厚度h也是常数，则式(6-29)可简化为

$$\boldsymbol{K}^e=\boldsymbol{B}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{B}hA \tag{6-30}$$

将应变矩阵式(6-23)和平面应力问题的弹性矩阵式(6-1)代入式(6-30)，则平面应力问题中常应变三角形单元的刚度矩阵的显式为

$$\boldsymbol{K}^e=\begin{bmatrix} \boldsymbol{K}_{ii} & \boldsymbol{K}_{ij} & \boldsymbol{K}_{im} \\ \boldsymbol{K}_{ji} & \boldsymbol{K}_{jj} & \boldsymbol{K}_{jm} \\ \boldsymbol{K}_{mi} & \boldsymbol{K}_{mj} & \boldsymbol{K}_{mm} \end{bmatrix} \tag{6-31}$$

其中

$$\boldsymbol{K}_{rs}=\boldsymbol{B}_r^{\mathrm{T}}\boldsymbol{D}\boldsymbol{B}_s hA=\frac{Eh}{4(1-\mu^2)A}\begin{bmatrix} b_rb_s+\dfrac{1-\mu}{2}c_rc_s & \mu b_rc_s+\dfrac{1-\mu}{2}c_rb_s \\ \mu c_rb_s+\dfrac{1-\mu}{2}b_rc_s & c_rc_s+\dfrac{1-\mu}{2}b_rb_s \end{bmatrix} \tag{6-32}$$

2) 等效节点力

(1) 单元自重. 设单元体积密度为ρ，g为重力加速度，自重沿y轴负向，则体积力向量可写为

$$\boldsymbol{p}_v=\begin{Bmatrix} 0 \\ -\rho g \end{Bmatrix} \tag{6-33}$$

则三角形单元的自重分配到节点上的等效节点力为

$$\boldsymbol{F}_v^e=\begin{Bmatrix} \boldsymbol{F}_{vi}^e \\ \boldsymbol{F}_{vj}^e \\ \boldsymbol{F}_{vm}^e \end{Bmatrix}=\iint_{\Delta}\begin{bmatrix} \boldsymbol{N}_i \\ \boldsymbol{N}_j \\ \boldsymbol{N}_m \end{bmatrix}\boldsymbol{p}_v h\mathrm{d}x\mathrm{d}y \tag{6-34}$$

其中

$$\boldsymbol{F}_{vi}^e=\iint_{\Delta}\boldsymbol{N}_i\boldsymbol{p}_v h\mathrm{d}x\mathrm{d}y \qquad (i,\ j,\ m) \tag{6-35}$$

则节点 i 的等效节点力为

$$F_{vi}^e=\begin{Bmatrix}F_{vix}\\F_{viy}\end{Bmatrix}=\iint_{\Delta}N_i\begin{Bmatrix}0\\-\rho g\end{Bmatrix}h\mathrm{d}x\mathrm{d}y=\begin{Bmatrix}0\\\frac{1}{3}\rho ghA\end{Bmatrix} \tag{6-36}$$

于是均质等厚度平面三角形单元自重引起的等效节点力为

$$F_v^e=-\frac{1}{3}\rho ghA\begin{bmatrix}1 & 0 & 1 & 0 & 1 & 0\end{bmatrix} \tag{6-37}$$

(2) 线性表面分布力. 设单元 ij 边长度为 l，受到线性分布的表面力 $p=[\sigma \quad \tau]^{\mathrm{T}}$，其中 σ 为法向压力，τ 为切向力，它们的正方向如图 6.4 所示. 取局部坐标 s，其原点为节点 i，由 i 指向 j 为正方向，则线性分布的压力表达式为

$$p=p_i+\frac{s}{l}(p_i-p_j) \tag{6-38}$$

故分布力向量为

$$p=\begin{Bmatrix}p_x\\p_y\end{Bmatrix}=\begin{Bmatrix}\sigma\sin\alpha-\tau\cos\alpha\\-\sigma\cos\alpha-\tau\sin\alpha\end{Bmatrix}=\frac{1}{l}\begin{Bmatrix}\sigma(y_i-y_j)-\tau(x_i-x_j)\\-\sigma(x_i-x_j)-\tau(y_i-y_j)\end{Bmatrix} \tag{6-39}$$

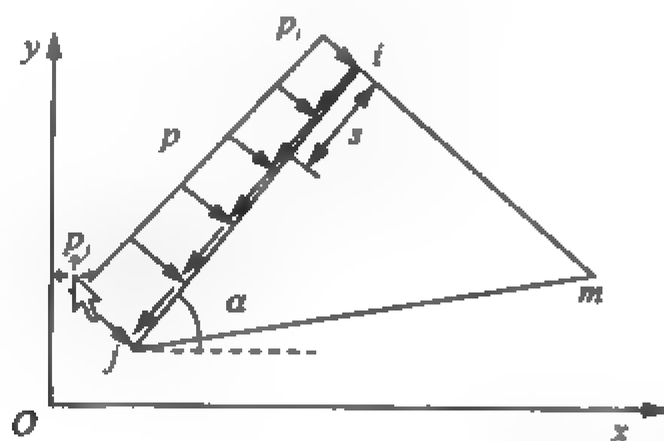


图 6.4 线性分布的压力

根据面积坐标的定义，用局部坐标 s 表示的 3 个形函数在 ij 边上的值为

$$N_i=1-\frac{s}{l},\quad N_j=\frac{s}{l},\quad N_m=0 \tag{6-40}$$

则线性分布表面力分配到节点上的等效节点力为

$$F_s^e=\begin{Bmatrix}F_{si}^e\\F_{sj}^e\\F_{sm}^e\end{Bmatrix}=\int_S\begin{bmatrix}N_i^{\mathrm{T}}\\N_j^{\mathrm{T}}\\N_j^{\mathrm{T}}\end{bmatrix}Ph\mathrm{d}s \tag{6-41}$$

将式(6-39)与式(6-40)代入式(6-41)得到

$$\begin{aligned}F_s^e=&\frac{1}{6}h\begin{bmatrix}(y_i-y_j)\sigma_1 & (x_j-x_i)\sigma_1 & (y_i-y_j)\sigma_2 & (x_j-x_i)\sigma_2 & 0 & 0\end{bmatrix}^{\mathrm{T}}\\&+\frac{1}{6}h\begin{bmatrix}(x_j-x_i)\tau_1 & (y_j-y_i)\tau_1 & (x_j-x_i)\tau_2 & (y_j-y_i)\tau_2 & 0 & 0\end{bmatrix}^{\mathrm{T}}\end{aligned} \tag{6-42}$$

其中

$$\sigma_1=2\sigma_i+\sigma_j,\quad \sigma_2=\sigma_i+2\sigma_j,\quad \tau_1=2\tau_i+\tau_j,\quad \tau_1=\tau_i+2\tau_j$$

式中，σ_i、σ_j、τ_i和τ_j是节点i和j的法向压力和切向压力分量的大小.

当表面力为均匀分布的压力时，即$\sigma_i=\sigma_j=\sigma_0$，则

$$F_s^e=\frac{1}{2}h\sigma_0\left[(y_i-y_j)\quad (x_j-x_i)\quad (y_i-y_j)\quad (x_j-x_i)\quad 0\quad 0\right]^{\mathrm{T}} \tag{6-43}$$

若表面力的方向固定，如表面载荷向量为

$$p=\begin{Bmatrix} p_x \\ p_y \end{Bmatrix}=\begin{Bmatrix} p_i+(p_j-p_i)s/l \\ 0 \end{Bmatrix} \tag{6-44}$$

则相应于式(6-44)的表面力的等效节点力为

$$F_s^e=\frac{1}{6}hl\left[(2p_i+p_j)\quad 0\quad (p_i+2p_j)\quad 0\quad 0\quad 0\right]^{\mathrm{T}} \tag{6-45}$$

进一步，当沿x方向的线性分布载荷为均匀分布载荷，即$p_i=p_j=p_0$时，等效节点力为

$$F_s^e=\frac{1}{2}p_0hl[1\quad 0\quad 1\quad 0\quad 0\quad 0]^{\mathrm{T}} \tag{6-46}$$

若为三角形分布载荷，此时$p_j=0$，则

$$F_s^e=\frac{1}{2}p_0hl\left[\frac{2}{3}\quad 0\quad \frac{1}{3}\quad 0\quad 0\quad 0\right] \tag{6-47}$$

5. 单元质量矩阵

1) 协调质量矩阵

设单元内ρ和h是常数，则有

$$M^e=\rho h\iint N^{\mathrm{T}}N\mathrm{d}x\mathrm{d}y \tag{6-48}$$

将式(6-19)代入式(6-48)得到

$$M^e=\rho h\iint\begin{bmatrix} N_i^2 & 0 & N_iN_j & 0 & N_iN_m & 0 \\ 0 & N_i^2 & 0 & N_iN_j & 0 & N_iN_m \\ N_jN_i & 0 & N_j^2 & 0 & N_jN_m & 0 \\ 0 & N_jN_i & 0 & N_j^2 & 0 & N_jN_m \\ N_mN_i & 0 & N_mN_j & 0 & N_m^2 & 0 \\ 0 & N_mN_i & 0 & N_mN_j & 0 & N_m^2 \end{bmatrix}\mathrm{d}x\mathrm{d}y \tag{6-49}$$

利用三角形面积坐标的积分公式，则有

$$\iint N_rN_s\mathrm{d}x\mathrm{d}y=\frac{1!1!0!}{(1+1+0+2)!}2A=\frac{1}{12}A\quad (r,s=i,j,m) \tag{6-50}$$

$$\iint N_r^2\mathrm{d}x\mathrm{d}y=\frac{2!0!0!}{(2+0+0+2)!}2A=\frac{1}{6}A\quad (r=i,j,m) \tag{6-51}$$

则基于式(6-50)和式(6-51)得到平面问题常应变三角形单元的协调质量矩阵(一致质量矩阵)为

$$M^e = \frac{\rho h A}{3}\begin{bmatrix} 1/2 & 0 & 1/4 & 0 & 1/4 & 0 \\ 0 & 1/2 & 0 & 1/4 & 0 & 1/4 \\ 1/4 & 0 & 1/2 & 0 & 1/4 & 0 \\ 0 & 1/4 & 0 & 1/2 & 0 & 1/4 \\ 1/4 & 0 & 1/4 & 0 & 1/2 & 0 \\ 0 & 1/4 & 0 & 1/4 & 0 & 1/2 \end{bmatrix} \tag{6-52}$$

2) 集中质量矩阵

将单元质量平均分配到单元的 3 个节点上便得到常应变三角形单元的集中质量矩阵

$$M^e = \frac{\rho h A}{3} I_6 \tag{6-53}$$

6.1.2 矩形双线性单元

1. 位移函数

如图 6.5(a)所示为 4 个节点的平面矩形单元，共有 8 个节点位移参数，可以直接在直角坐标系下进行分析. 为使分析过程简洁明了，这里引入一个无量纲的正则坐标(自然坐标)，令

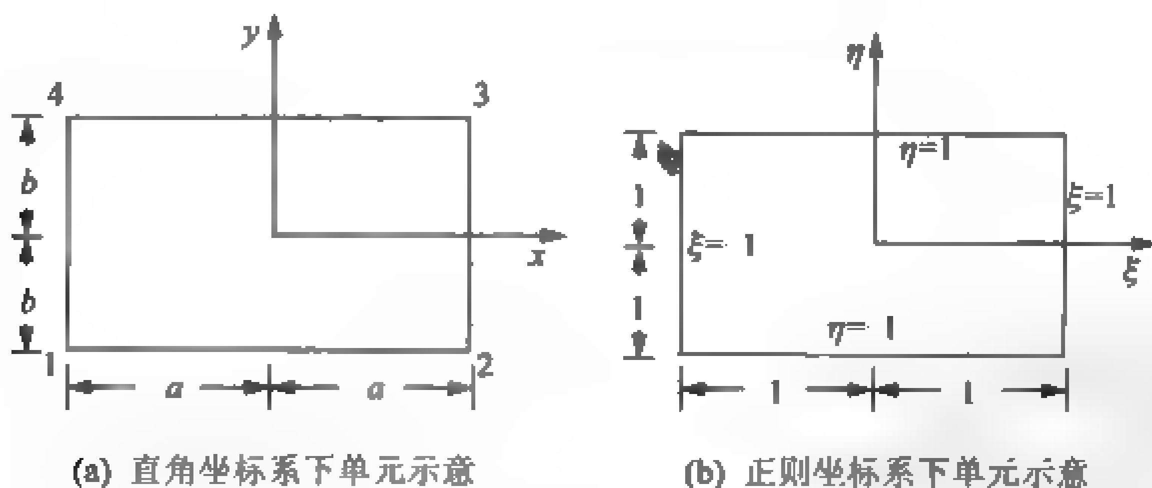
$$\xi = \frac{x}{a}, \quad \eta = \frac{y}{b} \tag{6-54}$$

在正则坐标系下原矩形单元映射为边长为 2 的正方形单元，如图 6.5(b)所示.

设单元的位移函数为

$$\begin{cases} u = \alpha_1 + \alpha_2\xi + \alpha_3\eta + \alpha_4\xi\eta \\ v = \alpha_5 + \alpha_6\xi + \alpha_7\eta + \alpha_8\xi\eta \end{cases} \tag{6-55}$$

式中，$\alpha_i\,(i=1, 2, \cdots, 8)$ 为广义坐标，可根据节点位移求出.



(a) 直角坐标系下单元示意　　(b) 正则坐标系下单元示意

图 6.5 矩形双线性单元

在 ξ, η 坐标下，单元的 4 条边界线的方程分别为

$$\begin{cases} \eta+1=0, \ \xi-1=0 \\ \eta-1=0, \ \xi+1=0 \end{cases} \tag{6-56}$$

根据形函数具有的性质：本点处形函数为 1，其他点处形函数为 0. 由式(6-56)可知下列函数

$$\begin{cases} N_1=\alpha(\xi-1)(\eta-1) \\ N_2=\beta(\xi+1)(\eta-1) \\ N_3=\gamma(\xi+1)(\eta+1) \\ N_2=\delta(\xi-1)(\eta+1) \end{cases} \tag{6-57}$$

将自动满足形函数其他点为零的性质. 代入本点坐标且令其等于1，可求得

$$\alpha=-\beta=\gamma=-\delta=\frac{1}{4} \tag{6-58}$$

将式(6-58)代回式(6-57)且引入如下记号

$$\xi_0=\xi_i\xi,\quad \eta_0=\eta_i\eta \tag{6-59}$$

则形函数可写为

$$N_i=(1+\xi_0)(1+\eta_0)/4 \tag{6-60}$$

单元的位移函数为

$$\boldsymbol{u}=\begin{Bmatrix} u \\ v \end{Bmatrix}=\boldsymbol{N}\boldsymbol{\delta}^e \tag{6-61}$$

其中

$$\boldsymbol{\delta}^e=[u_1\quad v_1\quad u_2\quad v_2\quad u_3\quad v_3\quad u_4\quad v_4]^{\mathrm{T}} \tag{6-62}$$

$$\boldsymbol{N}=[\boldsymbol{N}_1\quad \boldsymbol{N}_2\quad \boldsymbol{N}_3\quad \boldsymbol{N}_4]=[N_1\boldsymbol{I}_2\quad N_2\boldsymbol{I}_2\quad N_3\boldsymbol{I}_2\quad N_4\boldsymbol{I}_2] \tag{6-63}$$

2. 应变矩阵和应力矩阵

采用与三角形单元完全相同的分析方法和步骤可得以下结果.

1) 应变矩阵

$$\boldsymbol{B}=[\boldsymbol{B}_1\quad \boldsymbol{B}_2\quad \boldsymbol{B}_3\quad \boldsymbol{B}_4] \tag{6-64}$$

其中

$$\boldsymbol{B}_i=\frac{1}{4ab}\begin{bmatrix} b\xi_i(1+\eta_0) & 0 \\ 0 & a\eta_i(1+\xi_0) \\ a\eta_i(1+\xi_0) & b\xi_i(1+\eta_0) \end{bmatrix}\quad (i=1,2,3,4) \tag{6-65}$$

式中，a和b为x方向、y方向单元边长的一半.

2) 应力矩阵

平面应力的应力矩阵为

$$\boldsymbol{S}=[\boldsymbol{S}_1\quad \boldsymbol{S}_2\quad \boldsymbol{S}_3\quad \boldsymbol{S}_4] \tag{6-66}$$

其中

$$\boldsymbol{S}_i=\frac{E}{4ab(1-\nu^2)}\begin{bmatrix} b\xi_i(1+\eta_0) & \nu a\eta_i(1+\xi_0) \\ \nu b\xi_i(1+\eta_0) & a\eta_i(1+\xi_0) \\ \dfrac{1-\nu}{2}a\eta_i(1+\xi_0) & \dfrac{1-\nu}{2}b\xi_i(1+\eta_0) \end{bmatrix}\quad (i=1,2,3,4) \tag{6-67}$$

求解平面应变的应力矩阵，将式(6-67)中E改为$E/(1-\nu^2)$，ν换作$\nu/(1-\nu^2)$即可.

3. 单元刚度矩阵和单元等效载荷列阵

利用最小势能原理可推得单元的刚度矩阵和等效节点载荷列阵如下.

1) 单元刚度矩阵

$$\boldsymbol{K}^e=\begin{bmatrix}\boldsymbol{K}_{11} & \boldsymbol{K}_{12} & \boldsymbol{K}_{13} & \boldsymbol{K}_{14}\\ \boldsymbol{K}_{21} & \boldsymbol{K}_{22} & \boldsymbol{K}_{23} & \boldsymbol{K}_{24}\\ \boldsymbol{K}_{31} & \boldsymbol{K}_{32} & \boldsymbol{K}_{33} & \boldsymbol{K}_{34}\\ \boldsymbol{K}_{41} & \boldsymbol{K}_{42} & \boldsymbol{K}_{43} & \boldsymbol{K}_{44}\end{bmatrix} \tag{6-68}$$

其中

$$\boldsymbol{K}_{rs}^e=abt\int_{-1}^{1}\int_{-1}^{1}\boldsymbol{B}_r^{\mathrm{T}}\boldsymbol{D}\boldsymbol{B}_s\mathrm{d}\xi\mathrm{d}\eta \qquad (r,s=1,2,3,4) \tag{6-69}$$

平面应力时

$$\boldsymbol{K}_{rs}^e=\frac{Et}{4(1-\nu^2)ab}\begin{bmatrix}K_1 & K_2\\ K_3 & K_4\end{bmatrix} \qquad (r,s=1,2,3,4) \tag{6-70}$$

$$\begin{cases}K_1=b^2\xi_r\xi_s\left(1+\frac{1}{3}\eta_r\eta_s\right)+\frac{1-\nu}{2}a^2\eta_r\eta_s\left(1+\frac{1}{3}\xi_r\xi_s\right)\\ K_2=ab\left(\nu\eta_r\xi_s+\frac{1-\nu}{2}\xi_r\eta_s\right)\\ K_3=ab\left(\nu\xi_r\eta_s+\frac{1-\nu}{2}\eta_r\xi_s\right)\\ K_4=a^2\eta_r\eta_s\left(1+\frac{1}{3}\xi_r\xi_s\right)+\frac{1-\nu}{2}b^2\xi_r\xi_s\left(1+\frac{1}{3}\eta_r\eta_s\right)\end{cases} \tag{6-71}$$

平面应变时，将式(6-70)中 E 改为 $E/(1-\nu^2)$，ν 换作 $\nu/(1-\nu^2)$ 即可.

2) 单元等效节点载荷列阵

$$\boldsymbol{F}_E^e=t\left(\int_{-1}^{1}\boldsymbol{N}^{\mathrm{T}}\boldsymbol{F}_b^e\mathrm{d}\xi\mathrm{d}\eta+\sum\int_{L_e}\boldsymbol{N}^{\mathrm{T}}\boldsymbol{F}_s^e\mathrm{d}L\right) \tag{6-72}$$

式中后一项只有单元处于边界且受有表面力时才有.

4. 单元质量矩阵

1) 协调质量矩阵

矩形双线性单元的协调质量矩阵为

$$\boldsymbol{M}^e=\frac{\rho tA}{36}\begin{bmatrix}4&0&2&0&1&0&2&0\\0&4&0&2&0&1&0&2\\2&0&4&0&2&0&1&0\\0&2&0&4&0&2&0&1\\1&0&2&0&4&0&2&0\\0&1&0&2&0&4&0&2\\2&0&1&0&2&0&4&0\\0&2&0&1&0&2&0&4\end{bmatrix} \tag{6-73}$$

2) 集中质量矩阵

$$M^e = \frac{\rho t A}{4} I_6 \tag{6-74}$$

6.2 空间与轴对称问题

6.2.1 常应变四面体单元

1. 位移函数

如图 6.6 所示为常应变四面体单元，以 4 个顶点 i, j, k 和 l 为节点，每个节点有 3 个位移分量，可以写成列阵形式为

$$\delta_i = [u_i \quad v_i \quad w_i]^T \tag{6-75}$$

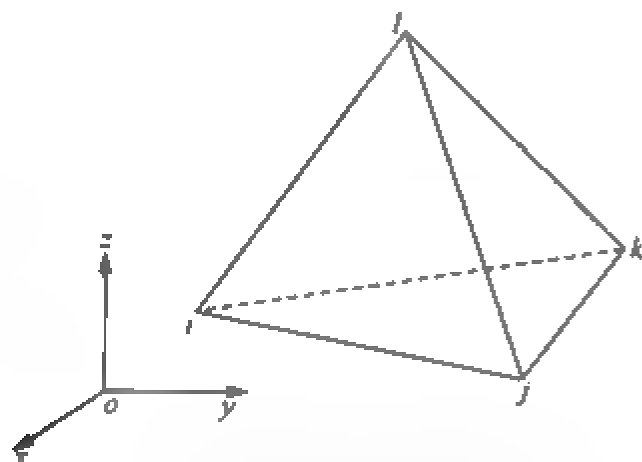


图 6.6 常应变四面体单元

因此，单元的节点位移矩阵为

$$\delta^e = \begin{bmatrix} \delta_1^T & \delta_2^T & \delta_3^T & \delta_4^T \end{bmatrix} \tag{6-76}$$

与平面问题相似，假定单元内任一点的位移分量是坐标的线性函数

$$\begin{cases} u = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 z \\ v = \alpha_5 + \alpha_6 x + \alpha_7 y + \alpha_8 z \\ w = \alpha_9 + \alpha_{10} x + \alpha_{11} y + \alpha_{12} z \end{cases} \tag{6-77}$$

设节点 i, j, k 和 l 的坐标分别为 (x_i, y_i, z_i), (x_j, y_j, z_j), (x_k, y_k, z_k) 和 (x_l, y_l, z_l)，把它们代入式(6-77)的第一式，得出各节点在 x 方向的位移是

$$\begin{Bmatrix} u_i \\ u_j \\ u_k \\ u_l \end{Bmatrix} = \begin{bmatrix} 1 & x_i & y_i & z_i \\ 1 & x_j & y_j & z_j \\ 1 & x_k & y_k & z_k \\ 1 & x_l & y_l & z_l \end{bmatrix} \begin{Bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{Bmatrix} \tag{6-78}$$

联立求解并代回整理即可写成与平面问题类似的形式：

$$u = \frac{1}{6V} \begin{bmatrix} (a_i + b_i x + c_i y + d_i z) u_i + (a_j + b_j x + c_j y + d_j z) u_j + \\ (a_k + b_k x + c_k y + d_k z) u_k + (a_l + b_l x + c_l y + d_l z) u_l \end{bmatrix} \tag{6-79}$$

其中

$$a_i=\begin{vmatrix} x_j & y_j & z_j \\ x_k & y_k & z_k \\ x_l & y_l & z_l \end{vmatrix},\quad b_i=-\begin{vmatrix} 1 & y_j & z_j \\ 1 & y_k & z_k \\ 1 & y_l & z_l \end{vmatrix},\quad c_i=\begin{vmatrix} 1 & x_j & z_j \\ 1 & x_k & z_k \\ 1 & x_l & z_l \end{vmatrix},$$

$$d_i=-\begin{vmatrix} 1 & x_j & y_j \\ 1 & x_k & y_k \\ 1 & x_l & y_l \end{vmatrix}\quad (i,j,k,l),\quad V=\frac{1}{6}\begin{vmatrix} 1 & x_i & y_i & z_i \\ 1 & x_j & y_j & z_j \\ 1 & x_k & y_k & z_k \\ 1 & x_l & y_l & z_l \end{vmatrix}$$

式中，V 是四面体的体积. 为了使四面体的体积不为负值，单元节点编号 i，j，k 和 l 必须按照一定的顺序. 在右手坐标系中，当按照 $i\to j\to k$ 方向转动时，右手螺旋应向 l 的方向前进.

引入如下形函数：

$$N_i=\frac{1}{6V}(a_i+b_ix+c_iy+d_iz)\qquad (i,j,k,l) \tag{6-80}$$

则式(6-78)也可写成

$$u=N_iu_i+N_ju_j+N_ku_k+N_lu_l \tag{6-81}$$

同理，另两个位移分量可表示为

$$v=N_iv_i+N_jv_j+N_kv_k+N_lv_l \tag{6-82}$$

$$w=N_iw_i+N_jw_j+N_kw_k+N_lw_l \tag{6-83}$$

则单元位移的矩阵表示为

$$\boldsymbol{u}=\begin{Bmatrix} u \\ v \\ w \end{Bmatrix}=\boldsymbol{N}\boldsymbol{\delta}^e=\begin{bmatrix} N_i\boldsymbol{I}_3 & N_j\boldsymbol{I}_3 & N_k\boldsymbol{I}_3 & N_l\boldsymbol{I}_3 \end{bmatrix}\boldsymbol{\delta}^e \tag{6-84}$$

2. 应变矩阵和应力矩阵

1) 应变矩阵

在空间问题中，每个点具有 6 个应变分量

$$\boldsymbol{\varepsilon}=\begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{Bmatrix}=\begin{Bmatrix} \dfrac{\partial u}{\partial x} \\ \dfrac{\partial v}{\partial y} \\ \dfrac{\partial w}{\partial z} \\ \dfrac{\partial u}{\partial y}+\dfrac{\partial v}{\partial x} \\ \dfrac{\partial v}{\partial z}+\dfrac{\partial w}{\partial y} \\ \dfrac{\partial w}{\partial x}+\dfrac{\partial u}{\partial z} \end{Bmatrix} \tag{6-85}$$

将式(6-84)代入式(6-85)可得

$$\boldsymbol{\varepsilon}=\boldsymbol{B}\boldsymbol{\delta}^{e}=\begin{bmatrix}\boldsymbol{B}_i & -\boldsymbol{B}_j & \boldsymbol{B}_k & -\boldsymbol{B}_l\end{bmatrix}\boldsymbol{\delta}^{e} \tag{6-86}$$

其中

$$\boldsymbol{B}_r=\frac{1}{6V}\begin{bmatrix}b_r & 0 & 0\\ 0 & c_r & 0\\ 0 & 0 & d_r\\ c_r & b_r & 0\\ 0 & d_r & c_r\\ d_r & 0 & b_r\end{bmatrix}\quad (r=i,j,k,l) \tag{6-87}$$

2) 应力矩阵

将式(6-86)代入物理方程可得

$$\boldsymbol{\sigma}=\begin{Bmatrix}\sigma_x & \sigma_y & \sigma_z & \tau_{xy} & \tau_{yz} & \tau_{zx}\end{Bmatrix}^{\mathrm{T}}=\boldsymbol{D}\boldsymbol{B}\boldsymbol{\delta}^{e}=\boldsymbol{S}\boldsymbol{\delta}^{e} \tag{6-88}$$

则应力矩阵 $\boldsymbol{S}$ 为

$$\boldsymbol{S}=\boldsymbol{D}\boldsymbol{B}=\begin{bmatrix}\boldsymbol{S}_i & \boldsymbol{S}_j & \boldsymbol{S}_k & \boldsymbol{S}_l\end{bmatrix} \tag{6-89}$$

其中

$$\boldsymbol{S}_r=\boldsymbol{D}\boldsymbol{B}_r=\frac{6A_3}{V}\begin{bmatrix}b_r & A_1c_r & A_1d_r\\ A_1b_r & c_r & A_1d_r\\ A_1b_r & A_1c_r & d_r\\ A_2c_r & A_2b_r & 0\\ 0 & A_2d_r & A_2c_r\\ A_2d_r & 0 & A_2b_r\end{bmatrix}\quad (r=i,j,k,l) \tag{6-90}$$

$$A_1=\frac{\nu}{1-\nu},\quad A_2=\frac{1-2\nu}{2(1-\nu)},\quad A_3=\frac{E(1-\nu)}{36(1+\nu)(1-2\nu)} \tag{6-91}$$

式中，对于各向同性材料，根据胡克定律，弹性矩阵 $\boldsymbol{D}$ 由下式决定：

$$\boldsymbol{D}=\frac{E(1-\nu)}{(1+\nu)(1-2\nu)}\begin{bmatrix}1 & \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 0 & 0 & 0\\ & 1 & \frac{\nu}{1-\nu} & 0 & 0 & 0\\ & & 1 & 0 & 0 & 0\\ & \text{对} & & \frac{1-2\nu}{2(1-\nu)} & 0 & 0\\ & & \text{称} & & \frac{1-2\nu}{2(1-\nu)} & 0\\ & & & & & \frac{1-2\nu}{2(1-\nu)}\end{bmatrix} \tag{6-92}$$

3. 单元刚度矩阵和等效节点载荷

1) 单元刚度矩阵

由最小势能原理可推得单元刚度矩阵为

$$K^e = \int_{V^e} B^T DB\mathrm{d}V = B^T DBV = \begin{bmatrix} K_{11} & K_{12} & K_{13} & K_{14} \\ K_{21} & K_{22} & K_{23} & K_{24} \\ K_{31} & K_{32} & K_{33} & K_{34} \\ K_{41} & K_{42} & K_{43} & K_{44} \end{bmatrix} \tag{6-93}$$

其中任一分块子矩阵 K_{rs} 为

$$\begin{aligned} K_{rs} &= B_r^T DB_s V \\ &= \frac{A_3}{V}\begin{bmatrix} b_r b_s + A_2(c_r c_s + d_r d_s) & A_1 b_r c_s + A_2 c_r b_s & A_1 b_r d_s + A_2 d_r b_s \\ A_1 c_r b_s + A_2 b_r c_s & c_r c_s + A_2(d_r d_s + b_r b_s) & A_1 c_r d_s + A_2 d_r c_s \\ A_1 d_r b_s + A_2 b_r d_s & A_1 d_r c_s + A_2 c_r b_s & d_r d_s + A_2(b_r b_s + c_r c_s) \end{bmatrix} \end{aligned} \tag{6-94}$$

2) 等效节点载荷

$$F_E^e = \int_V N^T F_b \mathrm{d}V + \int_{S_\sigma} N^T F_s \mathrm{d}S = \begin{bmatrix} F_{E1}^T & F_{E1}^T & F_{E1}^T & F_{E1}^T \end{bmatrix}^T \tag{6-95}$$

6.2.2　轴对称问题

工程中有一类结构，它们的几何形状、约束条件及作用的载荷都对称于某一固定轴，我们把它称为对称轴，则在载荷作用下结构所产生的位移、应变和应力也对称于该轴，这类问题称为**轴对称问题**. 若按空间问题对其进行分析，往往需要划分很多单元，因此未知量庞大，而利用问题的轴对称特点，则可将其转化为平面问题求解. 图 6.7 为烟囱柱坐标剖面单元离散示意图.

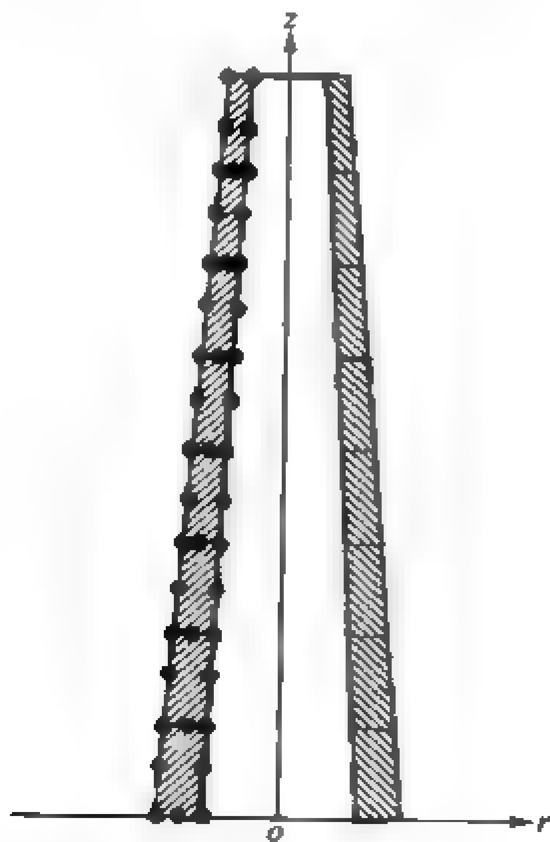


图 6.7　烟囱柱坐标剖面单元离散示意图

在轴对称问题中，通常采用圆柱坐标(r,θ,z). 以对称轴作为z轴，所有应力、应变和位移都与θ方向无关，只是r和z的函数，任一点的位移只有两个方向的分量，即沿r方向的径向位移u和沿z方向的轴向位移w. 由于轴对称，θ方向的位移v等于零，因此轴对称问题是二维问题.

若几何形状轴对称，载荷不是轴对称，则可以将载荷在θ方向展开成傅里叶级数，利用物体的对称性，可使问题的求解简化.

本节主要以三节点三角形环状单元为例进行讨论. 这种单元适应性好、计算简单，是一种常用的最简单的单元，其他单元有限元格式的建立，途径是一样的.

1. 位移函数

如图6.8所示，三角形环单元的单元位移场由前面所学内容可知

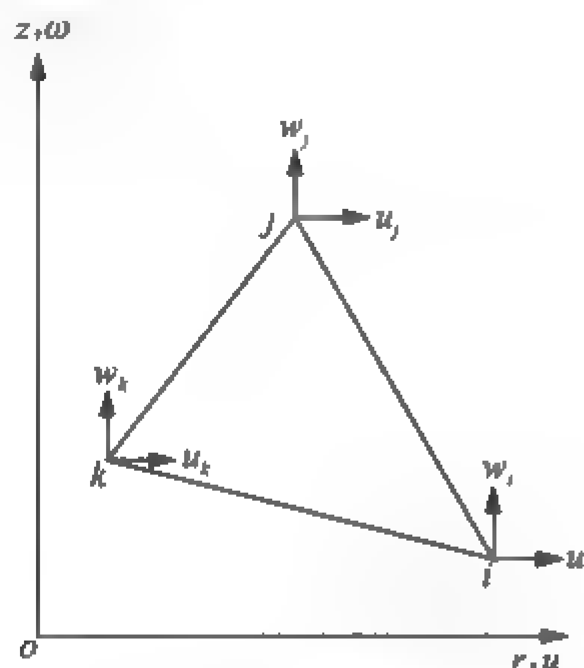


图6.8 三角形环单元

$$\boldsymbol{u}=[u\quad w]^{\mathrm{T}}=\left[N_i\boldsymbol{I}_2\quad N_j\boldsymbol{I}_2\quad N_k\boldsymbol{I}_2\right]\boldsymbol{\delta}^e \tag{6-96}$$

其中

$$N_i=\frac{1}{2A}(a_i+b_ir+c_iz)=L_i\quad(i,j,k)\quad(L_i\text{为面积坐标})$$

(是三角形环状单元截面积的2倍)

$$a_i=r_jz_k-r_kz_j,\quad b_i=z_j-z_k,\quad c_i=-(r_j-r_k)$$

2. 应变矩阵和应力矩阵

1) 应变矩阵

在柱坐标下弹性力学的几何方程为

$$\boldsymbol{\varepsilon}=\begin{Bmatrix}\varepsilon_r\\\varepsilon_z\\\gamma_{rz}\\\varepsilon_\theta\end{Bmatrix}=\begin{Bmatrix}\dfrac{\partial u}{\partial r}\\\dfrac{\partial w}{\partial z}\\\dfrac{\partial u}{\partial z}+\dfrac{\partial w}{\partial r}\\\dfrac{u}{r}\end{Bmatrix}=\begin{bmatrix}\dfrac{\partial}{\partial r}&0\\0&\dfrac{\partial}{\partial z}\\\dfrac{\partial}{\partial z}&\dfrac{\partial}{\partial r}\\\dfrac{1}{r}&0\end{bmatrix}\begin{Bmatrix}u\\w\end{Bmatrix}\tag{6-97}$$

将位移函数式(6-96)代入，得

$$\boldsymbol{\varepsilon}=\boldsymbol{B}\boldsymbol{\delta}^e=\begin{bmatrix}\boldsymbol{B}_i&\boldsymbol{B}_j&\boldsymbol{B}_k\end{bmatrix}\boldsymbol{\delta}^e\tag{6-98}$$

其中

$$\boldsymbol{B}_i=\frac{1}{2A}\begin{bmatrix}b_i&0\\0&c_i\\c_i&b_i\\f_i&0\end{bmatrix}\qquad(i,j,k,)$$

$$f_i=\frac{a_i}{r}+b_i+\frac{c_iz}{r}\qquad(i,j,k,)$$

可见三角形环单元与平面常应变单元是不同的，轴对称三角形环单元的应变矩阵现在不是常数矩阵.

2) 应力矩阵

单元应力可用应变代入弹性关系得到

$$\boldsymbol{\sigma}=\begin{Bmatrix}\sigma_r\\\sigma_z\\\tau_{rz}\\\sigma_\theta\end{Bmatrix}=\boldsymbol{D}\boldsymbol{\varepsilon}=\boldsymbol{D}\boldsymbol{B}\boldsymbol{\delta}^e=\boldsymbol{S}\boldsymbol{\delta}^e=\begin{bmatrix}\boldsymbol{S}_i&\boldsymbol{S}_j&\boldsymbol{S}_k\end{bmatrix}\boldsymbol{\delta}^e\tag{6-99}$$

式中，弹性矩阵 $\boldsymbol{D}$ 为

$$\boldsymbol{D}=\frac{E(1-\nu)}{(1+\nu)(1-2\nu)}\begin{bmatrix}1&\dfrac{\nu}{1-\nu}&\dfrac{\nu}{1-\nu}&0\\&1&\dfrac{\nu}{1-\nu}&0\\&\text{对}&1&0\\&&\text{称}&\dfrac{1-2\nu}{2(1-\nu)}\end{bmatrix}$$

应力矩阵 $\boldsymbol{S}$ 中子块 $\boldsymbol{S}_i$ 为

$$\boldsymbol{S}_i=\frac{E(1-\nu)}{2A(1+\nu)(1-2\nu)}\begin{bmatrix}b_i+A_1f_i&A_1c_i\\A_1(b_i+f_i)&c_i\\A_2c_i&A_2b_i\\A_1b_i+f_i&A_1c_i\end{bmatrix}\qquad(i,j,k,)$$

3. 单元刚度矩阵和等效节点载荷

1) 单元刚度矩阵

在轴对称的情况下，单元刚度矩阵为

$$\boldsymbol{K}^e=\iiint_{V^e}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{B}r\mathrm{d}\theta\mathrm{d}r\mathrm{d}z=2\pi\iint_{\Omega^e}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{B}r\mathrm{d}r\mathrm{d}z \tag{6-100}$$

为了简化计算和消除在对称轴上$r=0$所引起的麻烦，把单元中随点变化的r和z用单元截面形心处的坐标$\bar{r}$和$\bar{z}$来近似，即

$$r\approx\bar{r}=\frac{1}{3}(r_i+r_j+r_k),\quad z\approx\bar{z}=\frac{1}{3}(z_i+z_j+z_k) \tag{6-101}$$

则f_i近似为

$$f_i\approx\bar{f}_i=\frac{a_i}{\bar{r}}+b_i+\frac{c_i\bar{z}}{\bar{r}}\qquad (i,j,k,) \tag{6-102}$$

从而单元刚度矩阵的显式为

$$\boldsymbol{K}^e=2\pi\bar{r}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{B}A=\begin{bmatrix}\boldsymbol{K}_{ii}&\boldsymbol{K}_{ij}&\boldsymbol{K}_{ik}\\\boldsymbol{K}_{ji}&\boldsymbol{K}_{jj}&\boldsymbol{K}_{jk}\\\boldsymbol{K}_{ki}&\boldsymbol{K}_{kj}&\boldsymbol{K}_{kk}\end{bmatrix} \tag{6-103}$$

其中每一子块为

$$\boldsymbol{K}_{rs}=2\pi\bar{r}\boldsymbol{B}_r^{\mathrm{T}}\boldsymbol{D}\boldsymbol{B}_sA=\frac{\pi E(1-\nu)\bar{r}}{2A(1+\nu)(1-2\nu)}\begin{bmatrix}K_1&K_3\\K_2&K_4\end{bmatrix}$$

其中

$$\begin{cases}K_1=b_rb_s+f_rf_s+A_1(b_rf_s+f_rb_s)+A_2c_rc_s\\K_2=A_1c_r(b_s+f_s)+A_2b_rc_s\\K_3=A_1c_s(b_r+f_r)+A_2c_rb_s\\K_4=c_rc_s+A_2b_rb_s\end{cases}$$

2) 等效节点载荷

现在的等效节点载荷是由作用在单元面上的体积力、分布面力等引起的，对于轴对称问题为

$$\boldsymbol{F}_E^e=2\pi\int\boldsymbol{N}^{\mathrm{T}}\boldsymbol{F}_s r\mathrm{d}l+2\pi\int\boldsymbol{N}^{\mathrm{T}}\boldsymbol{F}_b r\mathrm{d}A$$

6.3 应用问题与MATLAB程序

【例 6.1】 如图 6.9 所示为受轴向载荷作用的板结构，其弹性模量为$6.9\times10^{10}\,\mathrm{Pa}$，泊松比为0.3，质量密度为$2\,700\mathrm{kg/m^3}$，单元厚度为0.001m．应用MATLAB程序求解结构在静载荷作用下的单元应力、节点位移和前10阶固有频率．

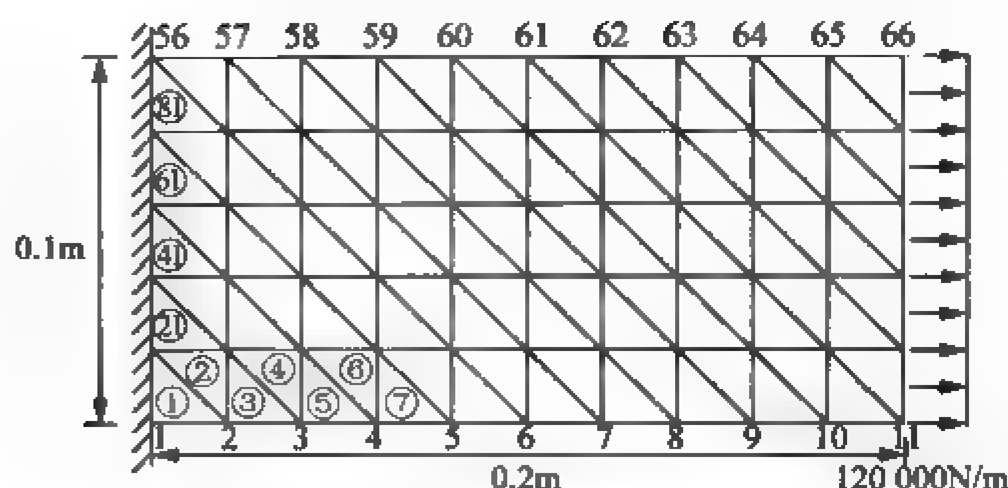


图 6.9　受轴向载荷的板结构

varargout=LineartriElement1(varargin)——计算线性三角形单元的质量矩阵和刚度矩阵. M 文件如下：

```
%--------------------------------------------------------------
% Example 6.1
% Dynamic characterisitic analysis and static analysis of the
% plate for plane stress analysis.
%
% Variable descriptions
%   dk = element stiffness matrix
%   dm = element mass matrix
%    k = system stiffness matrix
%    m = system mass matrix
%    gcoord = coordinate values of each node
%    nodes = nodal connectivity of each element
%--------------------------------------------------------------
% Input data for control paramters
%--------------------------------------------------------------

Element_number=100;       % number of elements
No_nel=3;                % number of nodes per element
No_dof=2;                % number of dofs per node
Node_number=66;          % total number of nodes in system
Prop(1)=6.9e10;           % elastic modulus
Prop(2)=0.3;             % Poisson's ratio
Prop(3)=2700;            % Mass density
t=0.001;                 % Thickness
%--------------------------------------------------------------
% Input data for nodal coordinate values
% gcoord(i,j) where i-> node no. and j-> x or y
%--------------------------------------------------------------
for loopi=1:Node_number
    if rem(loopi,11)~=0
        gcoord(loopi,1)=(loopi-floor(loopi/11)*11-1)*0.02;
        gcoord(loopi,2)=floor(loopi/11)*0.02;
```

```
      else
         gcoord(loopi,1)=0.2;
         gcoord(loopi,2)=(floor(loopi/11)-1)*0.02;
      end
  end
  %-------------------------------------------------------
  % Input data for nodal connectivity for each element
  % nodes(i,j) where i-> element no. and j-> connected nodes
  %-------------------------------------------------------
  No_element=0;
  for loopi=1:10
      for loopj=1:10
          No_element=No_element+1;
         if rem(No_element,2)~=0
            nodes(No element,1)=(No_element+1)/2+floor((loopi-1)/2);
            nodes(No_element,2)=(No_element+1)/2+1+floor((loopi-1)/2);
            nodes(No_element,3)=(No_element+1)/2+11+floor((loopi-1)/2);
         else
            nodes(No_element,1)=(No_element+2)/2+floor((loopi-1)/2);
            nodes(No_element,2)=(No_element+2)/2+11+floor((loopi-1)/2);
            nodes(No_element,3)=(No_element+2)/2+10+floor((loopi-1)/2);
         end
      end
  end
  %-------------------------------------------------------
  % Inout data for boundary conditions
  %-------------------------------------------------------
  ed(1:Node_number,1:2)=1;           %element_displacement
  constraint=[1,1;1,2;12,1;12,2;23,1;23,2;34,1;34,2;45,1;45,2;56,1;56,2];

  for loopi=1:length(constraint);
     ed(constraint(loopi,1),constraint(loopi,2))=0;
  end
  dof=0;
  for loopi=1:Node_number
      for loopj=1:2
         if ed(loopi,loopj)~=0
             dof=dof+1;
             ed(loopi,loopj)=dof;
         end
     end
  end
  %-------------------------------------------------------
  %Initilization of matrices
  %-------------------------------------------------------
  k=zeros(dof,dof);           % system stiffness matrix
  m=zeros(dof,dof);           % system mass matrix
```

```
f=zeros(Node_number*No_dof,1); % system force vector
disp-zeros(dof,1);        % system displacement vector
eldisp=zeros(No_nel*No_dof,1); % element diaplacement vector
stress=zeros(Element_number,3); % matrix containing stress components
strain-zeros(Element_number,3); % matrix containing strain components
kinmtx zeros(3,No_nel*No_dof);  % kinematic matrix
matmtrx=zeros(3,3);             % constitutive matrix
e2s(1:6)=0;    % index of transform the element displament number to
structural
%-------------------
% force vector
%-------------------- -   --
f(21)=5000;
f(131)=5000;
%---          ----------------------     --------------
% Compute system stiffness and mass matrices
%         -------------------- -       ------------------
 for loopi-1:Element_number           % loop for the total number of
elements
    for zi=1:3
      e2s((zi-1)*2+1)=ed(nodes(loopi,zi),1);
      e2s((zi-1)*2+2)=ed(nodes(loopi,zi),2);
    end
   for loopj=1:3
      xycoord(loopj,1)=gcoord(nodes(loopi,loopj),1);
      xycoord(loopj,2)=gcoord(nodes(loopi,loopj),2);
   end
 iopt=1;                             % plane stress analysis
% iopt=2;                            % plane strain analysis
 Opt_mass-1;                         % Consistent mass matrix
 [dk,dm]=LineartriElement1(Prop,No nel,No_dof,xycoord,t,iopt,Opt_mass);
   for jx-1:6
      for jy=1:6
         if(e2s(jx)*e2s(jy)~=0)
            k(e2s(jx),e2s(jy))=k(e2s(jx),e2s(jy))+dk(jx,jy);
            m(e2s(jx),e2s(jy))=m(e2s(jx),e2s(jy))+dm(jx,jy);
         end
      end
   end
end
%------ -------- --------
% compute system displacement
%------- - - ---
Number_con=-1;                % apply boundary conditions
for loopi=1:length(constraint)
   Number_con= Number_con+1;
   f((constraint(loopi,1) 1)*No_dof+constraint(loopi,2)-Number_con)=[];
```

```
end
disp=k\f;                    %solve the matrix equation
%-------------------------------
% element stress and strain computation
%-------------------------------
for loopi=1:Element_number
    for zi=1:3                        % extract system dofs for the element
        e2s((zi-1)*2+1)=ed(nodes(loopi,zi),1);
        e2s((zi-1)*2+2)=ed(nodes(loopi,zi),2);
    end
    for loopj=1:3
       xycoord(loopj,1)=gcoord(nodes(loopi,loopj),1);
       xycoord(loopj,2)=gcoord(nodes(loopi,loopj),2);
    end
    for loopk=1:No_nel*No dof % extract element displacement vector
        if e2s(loopk)==0
           eldisp(loopk)=0;
        else
           eldisp(loopk)=disp(e2s(loopk));
        end
    end

     x1=xycoord(1,1);y1=xycoord(1,2);
     x2=xycoord(2,1);y2=xycoord(2,2);
     x3=xycoord(3,1);y3=xycoord(3,2);
     area=0.5*(x1*y2+x2*y3+x3*y1 x1*y3-x2*y1 x3*y2);  % area of triangule
     dhdx=(1/(2*area))*[(y2-y3) (y3-y1) (y1-y2)];  % derivatives w.r.t x
     dhdy=(1/(2*area))*[(x3-x2) (x1-x3) (x2-x1)];  % derivatives w.r.t y
     for i=1:No_nel                                 % kinematic matrix
         i1=(i-1)*2+1;
         i2=i1+1;
         kinmtx(1,i1)=dhdx(i);
         kinmtx(2,i2)=dhdy(i);
         kinmtx(3,i1)=dhdy(i);
         kinmtx(3,i2)=dhdx(i);
     end
      matmtrx=Prop(1)/(1-Prop(2)*Prop(2))* ...
                 [1 Prop(2) 0; ...
                 Prop(2) 1 0;  ...
                 0 0 (1-Prop(2))/2];
      estrain=kinmtx*eldisp;                % compute strains
      estress=matmtrx*estrain ;              % compute stresses
      for i=1:3
          strain(loopi,i)=estrain(i);       % store for each element
          stress(loopi,i)=estress(i);       % store for each element
      end
```

```
end
%------------------------------------------------------------
% Construct the taotal displacement
%------------------------------------------------------------
flag=1;
iter=1;
displ=disp;
while flag
    for loopi=1:length(constraint)
        iter=iter+1;
        for loopj=1:(dof+iter)
            if loopj<((constraint(loopi,1)-1)*No_dof+constraint(loopi,2))
                dispt(loopj)=displ(loopj);
            else
                dispt(loopj+1)=displ(loopj);
            end
        end
        dispt((constraint(loopi,1)-1)*No_dof+constraint(loopi,2))=0;
        displ=dispt;

    end
    if length(displ)>=Node_number*No_dof
            flag=0;
    end
end
%--------------------------------------------------
% Compute the first ten natrual frequencies
%--------------------------------------------------
p=10;
epsilon=1e-5;
[v,d]=Ssiter(k,m,p,epsilon);
d=sqrt(d)/(2*pi);
%------------------------------------------------------------
%  print fem solutions
%------------------------------------------------------------
 num=1:1:p;
 frequency=[num' d]                % print frequency
%
 num=1:1:Node_number*No_dof;
 displace=[num' dispt']            % print nodal displacements
%
 num=1:Element_number;
stresses=[num' stress]             % print stresses

>> format long

frequency
```

```
  1.0e+004 *
   0.00010000000000   0.18604065704136
   0.00020000000000   0.63693976247305
   0.00030000000000   0.70795736724809
   0.00040000000000   1.49743000911713
   0.00050000000000   1.87501877728104
   0.00060000000000   1.88424652438432
   0.00070000000000   2.42279455024536
   0.00080000000000   2.54380218643713
   0.00090000000000   2.69029118728467
   0.00100000000000   2.71490380740973
displace =
  1.0e+002 *
   0.01000000000000                  0
   0.02000000000000                  0
   0.03000000000000   0.00000034291373
   0.04000000000000   0.00000015963419
   ......
   1.29000000000000   0.00000327251912
   1.30000000000000  -0.00000032693461
   1.31000000000000   0.00000380946666
   1.32000000000000  -0.00000054632754
stresses =
  1.0e+008 *
   0.00000001000000  1.30005754768989  0.39001726430697  0.21182228466200
   0.00000002000000  1.12293677324908  0.07616439071518  0.07963707173014
   0.00000003000000  1.17720848679054  0.09244590477762  0.04321198916920
   0.00000004000000  1.17304599196842  0.02612618266368  0.02423600768642
   ......
   0.00000097000000  1.17311067119685  0.00959355874785  0.06554658062287
   0.00000098000000  1.52792274071915  0.07363651493596  0.17831524139106
   0.00000099000000  1.14515705809605 -0.04119318985097  0.08300345141375
   0.00000100000000  1.78924143352395 -0.21075856647605  0.21075856647604

function varargout=LineartriElement1(varargin)
%----------------------------------------------------------------------
% Purpose:
% This function is used to calculate element stiffness and mass matrixes
% of linear triangular element
% Synopsis:
%     k=LineartriElement1(Prop, No_nel,No_dof,xycoord,thickness,iopt)
%     [k,m] LineartriElement1(Prop, No_nel,No_dof,xycoord,thickness,iopt,Opt_mass)
% Variable Description:
%       Input parameters
%          Prop(1)- elastic modulus
%          Prop(2)- Poisson's ratio
%          Prop(3)- mass density
```

```
%         No_nel - number of nodes per element
%         No_dof - number of dofs per node
%         xycoord -coord values of nodes
%         iopt=1 - plane stress analysis
%         iopt=2 - plane strain analysis
%         thickness - element thickness
%         Opt_mass =1 - consistent mass matrix
%         Opt_mass= 2 - lumped mass matrix
%      Output parameters
%          k - element stiffness matrix
%          m - element mass matrix
% Author: Dr.XU Bin, Time:2006-12-08
%---------------------------------------------------------------------

if nargin<6 & nargin>7
   error('Incorrect number of input arguments')
else
  switch nargin
     case 6
        Prop=varargin{1};
        No_nel=varargin{2};
        No_dof=varargin{3};
        xycoord=varargin{4};
        thickness=varargin{5};
        iopt=varargin{6};
        k=zeros(No_nel*No_dof);
        kinmtx=zeros(3,No_nel*No_dof);
        matmtrx=zeros(3,3);
 %-----------------------------------------------------------------------
 % the constitutive equation for isotropic material
 %-----------------------------------------------------------------------
        if iopt==1             % constitutive matrix for plane stress
           matmtrx=Prop(1)/(1-Prop(2)*Prop(2))* ...
               [1 Prop(2) 0; ...
               Prop(2) 1 0;  ...
               0 0 (1-Prop(2))/2];
        else
           matmtrx=Prop(1)/((1+Prop(2))*(1-2*Prop(2)))* ...
           % constitutive matrix for plane strain
               [1-Prop(2) Prop(2) 0; ...
               Prop(2) 1-Prop(2) 0;  ...
               0 0 (1-2*Prop(2))/2];
        end
%-----------------------------------------------------------------------
% the kinematic equation between strains and displacements
%-----------------------------------------------------------------------
        x1=xycoord(1,1);y1=xycoord(1,2);
```

```
        x2=xycoord(2,1);y2=xycoord(2,2);
        x3=xycoord(3,1);y3=xycoord(3,2);
        area=0.5*(x1*y2+x2*y3+x3*y1-x1*y3-x2*y1-x3*y2); % area of triangule
        dhdx=(1/(2*area))*[(y2-y3) (y3 y1) (y1-y2)]; % derivatives w.r.t x
        dhdy=(1/(2*area))*[(x3 x2) (x1-x3) (x2-x1)]; % derivatives w.r.t y
        for i=1:No_nel
            i1=(i-1)*2+1;
            i2=i1+1;
            kinmtx(1,i1)=dhdx(i);
            kinmtx(2,i2)=dhdy(i);
            kinmtx(3,i1)=dhdy(i);
            kinmtx(3,i2)=dhdx(i);
        end
%---------------------------------------------------------------------
% element stiffness matrix
%---------------------------------------------------------------------
  k=kinmtx'*matmtrx*kinmtx*area*thickness;

%----------------------------------------
% output element stiffness matrix
%----------------------------------------
  varargout(1)=k;
%----------------------------------------

    case 7
        Prop=varargin(1);
        No_nel=varargin(2);
        No_dof=varargin(3);
        xycoord=varargin(4);
        thickness=varargin(5);
        iopt=varargin(6);
        Opt_mass=varargin(7);
        k=zeros(No_nel*No_dof);
        m=zeros(No_nel*No_dof);
        kinmtx=zeros(3,No_nel*No_dof);
        matmtrx=zeros(3,3);
%---------------------------------------------------------------------
% the constitutive equation for isotropic material
%---------------------------------------------------------------------
        if iopt==1           % constitutive matrix for plane stress
            matmtrx=Prop(1)/(1-Prop(2)*Prop(2))* ...
                [1 Prop(2) 0; ...Prop(2)
                Prop(2) 1 0;  ...
                0 0 (1-Prop(2))/2];
        else
            matmtrx=Prop(1)/((1+Prop(2))*(1-2*Prop(2)))* ...
            % constitutive matrix for plane strain
```

```
                [1 Prop(2) Prop(2) 0; ...
                Prop(2) 1-Prop(2) 0;  ...
                0 0 (1-2*Prop(2))/2];
        end
%       ----------------------------      ---------------
% the kinematic equation between strains and displacements
%   ----------------------         ---------------      -----------     -
        x1=xycoord(1,1);y1=xycoord(1,2);
        x2=xycoord(2,1);y2=xycoord(2,2);
        x3=xycoord(3,1);y3=xycoord(3,2);
        area=0.5*(x1*y2+x2*y3+x3*y1-x1*y3-x2*y1-x3*y2); % area of triangule
        dhdx=(1/(2*area))*[(y2-y3) (y3-y1) (y1-y2)]; % derivatives w.r.t x
        dhdy=(1/(2*area))*[(x3-x2) (x1-x3) (x2-x1)]; % derivatives w.r.t y
        for i=1:No nel
            i1=(i-1)*2+1;
            i2=i1+1;
            kinmtx(1,i1)=dhdx(i);
            kinmtx(2,i2)=dhdy(i);
            kinmtx(3,i1)=dhdy(i);
            kinmtx(3,i2)=dhdx(i);
        end
%---------------       ------------     --------------------------------
% element stiffness matrix
%-------------------------------------------------     ---------    -------
  k=kinmtx'*matmtrx*kinmtx*area*thickness;

%--------------------------------------------------------------------------
% element mass matrix
%-        ------------      ----------------------------------------------

if Opt_mass==1                                   % consistent mass matrix
   m=Prop(3)*thickness*area/12*[2 0 1 0 1 0; ...
       0 2 0 1 0 1; ...
       1 0 2 0 1 0; ...
       0 1 0 2 0 1; ...
       1 0 1 0 2 0; ...
       0 1 0 1 0 2];
else                                           % lumped mass matrix
   m=Prop(3)*thickness*area/3*eye(6);
end
%          ----- --    --------     --------    -------    -
% output element stiffness matrix and element mass matrix
%-------     --------        ----         ------     ------
   varargout{1}=k;
   varargout{2}=m;
  end
end
```

```
%-----------------------------------------------------------
% The end
%-----------------------------------------------------------
```

【例 6.2】 如图 6.10 所示为平面应力板，其弹性模量为2.1×10^{11}Pa， 泊松比为 0.3，质量密度为7860kg/m^3，单元厚度为0.003m．在节点 6 处加一竖直向下、大小为 2000N 的阶跃力，则该点竖直方向阶跃响应如图 6.11 所示.

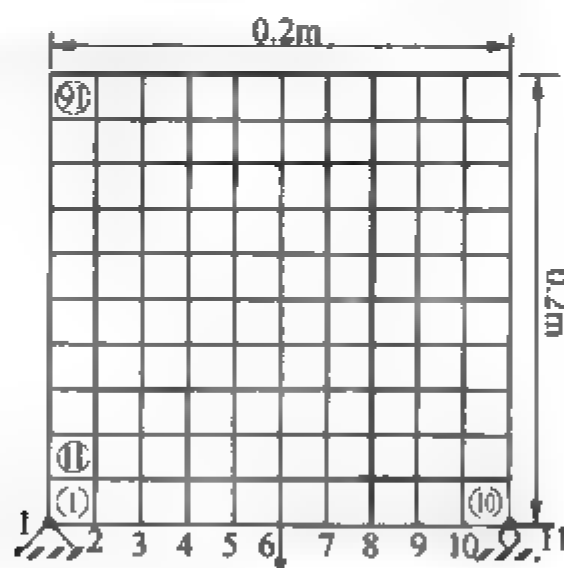


图 6.10 平面应力板

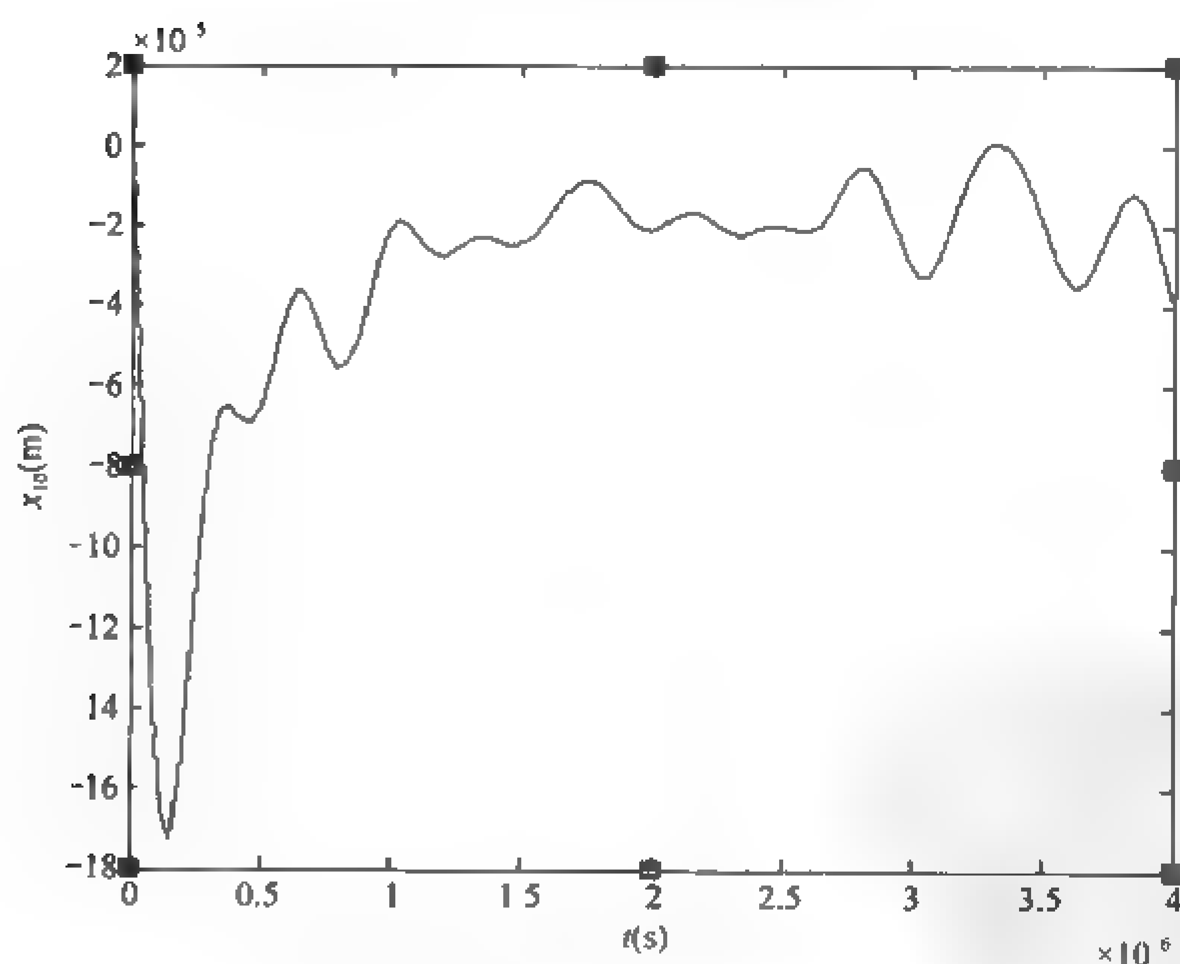


图 6.11 节点 6 竖直方向的阶跃响应

varargout=BilinearElement1(varargin)　计算双线性三角形单元的质量矩阵和刚度矩阵.

M 文件如下：

```
%   ----     -----    ------    -------      - ----
% Example 6.2
% Dynamic resposne analysis of the plate for plane stress
% analysis.
%
% Variable descriptions
%   dk = element stiffness matrix
%   dm = element mass matrix
%    k = system stiffness matrix
%    m = system mass matrix
%    gcoord = coordinate values of each node
%    nodes = nodal connectivity of each element
%------- ---- ---- -------------------
% Input data for control paramters
%------  -----  ----- ------------------------ -----
Element_number=100;       % number of elements
No_nel=4;                 % number of nodes per element
No_dof=2;                 % number of dofs per node
Node_number=121;          % total number of nodes in system
Prop(1)=2.1e11;           % elastic modulus
Prop(2)=0.3;              % Poisson's ratio
Prop(3)=7860;             % Mass density
t=0.003;                  % Thickness
%-------------------------------------------------------------
% Input data for nodal coordinate values
% gcoord(i,j) where i-> node no. and j-> x or y
%------------------------------ --------  ------  --------

for loopi=1:Node_number
    if rem(loopi,11)~=0
        gcoord(loopi,1)=(loopi-floor(loopi/11)*11-1)*0.02;
        gcoord(loopi,2)=floor(loopi/11)*0.02;
    else
        gcoord(loopi,1)=0.2;
        gcoord(loopi,2)=(floor(loopi/11)-1)*0.02;
    end
end
%---  -----  ------  ------ ----- ---------  ----  ---
% Input data for nodal connectivity for each element
% nodes(i,j) where i-> element no. and j-> connected nodes
% ----  --  ----  -----  ----  -----  ---- ------
No_element=0;
for loopi=1:10
    for loopj=1:10
        No_element=No_element+1;
        nodes(No_element,1)=(loopi-1)*11+loopj;
```

```
            nodes(No_element,2)=(loopi-1)*11+loopj+1;
            nodes(No_element,3)=(loopi-1)*11+loopj+12;
            nodes(No_element,4)=(loopi-1)*11+loopj+11;
    end
end
%------------------------------------------------------------
% Inout data for boundary conditions
%------------------------------------------------------------
ed(1:Node_number,1:2)=1;           %element_displacement
constraint=[1,1;1,2;11,2];

for loopi=1:length(constraint);
    ed(constraint(loopi,1),constraint(loopi,2))=0;
end
dof=0;
for loopi=1:Node_number
     for loopj=1:2
        if ed(loopi,loopj)~=0
            dof=dof+1;
            ed(loopi,loopj)=dof;
        end
    end
end
%------------------------------------------------------------
%Initilization of matrices
%------------------------------------------------------------
k=zeros(dof,dof);          % system stiffness matrix
m=zeros(dof,dof);          % system mass matrix
e2s(1:8)=0;    % index of transform the element displament number to
structural
%------------------------------------------------------------
% Compute system stiffness and mass matrices
%------------------------------------------------------------
 for loopi=1:Element_number   %  loop for the total number of elements
     for zi=1:4
        e2s((zi-1)*2+1)=ed(nodes(loopi,zi),1);
        e2s((zi-1)*2+2)=ed(nodes(loopi,zi),2);
     end
   for loopj=1:4
       xycoord(loopj,1)=gcoord(nodes(loopi,loopj),1);
       xycoord(loopj,2)=gcoord(nodes(loopi,loopj),2);
   end
  iopt=1;                          % plane stress analysis
 % iopt=2;                         % plane strain analysis
  Opt mass=1;                      % Consistent mass matrix
  [dk,dm]=BilinearElement1(Prop, No_nel,No dof,xycoord,t,iopt,Opt_mass);
    for jx=1:8
```

```
        for jy=1:8
           if(e2s(jx)*e2s(jy)~=0)
              k(e2s(jx),e2s(jy))=k(e2s(jx),e2s(jy))+dk(jx,jy);
              m(e2s(jx),e2s(jy))=m(e2s(jx),e2s(jy))+dm(jx,jy);
           end
        end
     end
  end
%-------------------------------------------------------------
% Dynamic response analysis
%-------------------------------------------------------------
A=[zeros(dof) eye(dof);-inv(m)*k zeros(dof)];  % state-space form
B=zeros(2*dof,1);
B(10)=-2000;
C=eye(2*dof);
D=0;
G=ss(A,B,C,D);
t=0:1e-8:4e-6
[y,t,x]=step(G);
plot(t,x(:,10))
%-------------------------------------------------------------
% The end
%-------------------------------------------------------------
```

```
function varargout=BilinearElement1(varargin)
%--------------------------------------------------------------------------
% Purpose:
% This function is used to calculate element stiffness and mass matrixes
% of bilinear triangular element
% Synopsis:
%     k=BilinearElement11(Prop, No_nel,No_dof,gcoord,thickness,iopt)
%     [k,m]=BilinearElement1(Prop, No_nel,No_dof,gcoord,thickness,iopt,Opt_mass)
% Variable Description:
%      Input parameters
%         Prop(1)- elastic modulus
%         Prop(2)- Poisson's ratio
%         Prop(3)- mass density
%         No_nel - number of nodes per element
%         No_dof - number of dofs per node
%         xycoord -coord values of nodes
%         iopt=1 - plane stress analysis
%         iopt=2 - plane strain analysis
%         thickness - element thickness
%         Opt_mass =1 - consistent mass matrix
%         Opt_mass= 2 - lumped mass matrix
```

```
%      Output parameters
%          k - element stiffness matrix
%          m - element mass matrix
% Author: Dr.XU Bin, Time:2006-12-08
%-     ----- ---     ----------      --------------      -------------
if nargin<6 |nargin>7
   error('Incorrect number of input arguments')
else
   switch nargin
      case 6
         Prop=varargin{1};
         No_nel=varargin{2};
         No_dof=varargin{3};
         xycoord=varargin{4};
         thickness=varargin{5};
         iopt=varargin{6};
         nglx=2; ngly=2;        % 2*2 Gauss-Legendre quadrature
         nglxy=nglx*ngly;       % number of sampling points per element
         k=zeros(No_nel*No_dof);
         kinmtx=zeros(3,No_nel*No_dof);
         matmtrx=zeros(3,3);
%-------------------         ------------------------------------- --
% the constitutive equation for isotropic material
%---------------------------------------------------------------------------
         if iopt==1              % constitutive matrix for plane stress
            matmtrx=Prop(1)/(1-Prop(2)*Prop(2))* ...
               [1 Prop(2) 0; ...
               Prop(2) 1 0;  ...
               0 0 (1-Prop(2))/2];
         else
            matmtrx=Prop(1)/((1+Prop(2))*(1-2*Prop(2)))* ...
            % constitutive matrix for plane strain
               [1-Prop(2) Prop(2) 0; ...
               Prop(2) 1-Prop(2) 0;  ...
               0 0 (1-2*Prop(2))/2];
         end
%--------------------------------------------------------------------------
% numerical integratrion for  element stiffness matrix
%-------------------------------------------------------------------
      point(1)=-0.577350269189626;        % Sampling pointrs & weights
      point(2)=-point(1);
      weight(1)=1.0;
      weight(2)=weight(1);
      for i=1:No_nel
         xcoord(i)=xycoord(i,1);           % extract x value of the node
         ycoord(i)=xycoord(i,2);           % extract y value of the node
      end
```

```
        for intx=1:nglx
           x=point(intx);                          % sampling point in x-axis
           wtx=weight(intx);                        %  weight in x-axis
           for inty=1:ngly
              y=point(inty);                        % sampling point in y-axis
              wty=weight(inty);                    % weight in y-axis
                  dhdr(1)=-0.25*(1-y);
                  % Compute the derivatives of shape functions
                  dhdr(2)=0.25*(1-y);              % at sampling point
                  dhdr(3)=0.25*(1+y);
                  dhdr(4)=-0.25*(1+y);
                  dhds(1)=-0.25*(1-x);
                  dhds(2)=-0.25*(1+x);
                  dhds(3)=0.25*(1+x);
                  dhds(4)=0.25*(1-x);

          jacob=zeros(2,2);
          for i=1:No_nel                            % compute Jacobian
             jacob(1,1)=jacob(1,1)+dhdr(i)*xcoord(i);
             jacob(1,2)=jacob(1,2)+dhdr(i)*ycoord(i);
             jacob(2,1)=jacob(2,1)+dhds(i)*xcoord(i);
             jacob(2,2)=jacob(2,2)+dhds(i)*ycoord(i);
          end
          detjacob=det(jacob);         % determinant of Jacobian
          invjacob=inv(jacob);         % inverse of Jacobian matrix
           for i=1:No_nel
             dhdx(i)=invjacob(1,1)*dhdr(i)+invjacob(1,2)*dhds(i);
             dhdy(i)=invjacob(2,1)*dhdr(i)+invjacob(2,2)*dhds(i);
           end
          for i=1:No_nel
             i1=(i-1)*2+1;
             i2=i1+1;
             kinmtx(1,i1)=dhdx(i);
             kinmtx(2,i2)=dhdy(i);
             kinmtx(3,i1)=dhdy(i);
             kinmtx(3,i2)=dhdx(i);
          end

            k=k+kinmtx'*matmtrx*kinmtx*wtx*wty*detjacob*thickness;
           end
        end
%   --------    ------    ------     ------
% output element stiffness matrix
%-    -------    ---- -----    -----    ---
  varargout{1}=k;

     case 7
```

```
        Prop=varargin{1};
        No_nel=varargin{2};
        No_dof=varargin{3};
        xycoord=varargin{4};
        thickness=varargin{5};
        iopt=varargin{6};
        Opt_mass=varargin{7};
        nglx=2; ngly=2;          % 2*2 Gauss-Legendre quadrature
        nglxy=nglx*ngly;         % number of sampling points per element
        k=zeros(No_nel*No_dof);
        m=zeros(No_nel*No_dof);
        kinmtx=zeros(3,No_nel*No_dof);
        matmtrx=zeros(3,3);
%-----------------------------------------------------------------------
% the constitutive equation for isotropic material
%-----------------------------------------------------------------------
        if iopt==1             % constitutive matrix for plane stress
           matmtrx=Prop(1)/(1-Prop(2)*Prop(2))* ...
               [1 Prop(2) 0; ...Prop(2)
               Prop(2) 1 0;  ...
               0 0 (1-Prop(2))/2];
        else
           matmtrx=Prop(1)/((1+Prop(2))*(1-2*Prop(2)))* ...
           % constitutive matrix for plane strain
               [1-Prop(2) Prop(2) 0; ...
               Prop(2) 1-Prop(2) 0;  ...
               0 0 (1-2*Prop(2))/2];
        end
%-----------------------------------------------------------------------
% numerical integratrion for  element stiffness matrix
%-----------------------------------------------------------------------

      point(1)=-0.577350269189626;      % Sampling pointrs & weights
      point(2)=-point(1);
      weight(1)=1.0;
      weight(2)=weight(1);
      for i=1:No_nel
         xcoord(i)=xycoord(i,1);       % extract x value of the node
         ycoord(i)=xycoord(i,2);       % extract y value of the node
      end
      for intx=1:nglx
         x=point(intx);                     % sampling point in x-axis
         wtx=weight(intx);                  %  weight in x-axis
         for inty=1:ngly
            y=point(inty);                  % sampling point in y-axis
            wty=weight(inty);               % weight in y-axis
                dhdr(1)=-0.25*(1-y);
```

```
                % Compute the derivatives of shape functions
                dhdr(2)=0.25*(1-y);          % at sampling point
                dhdr(3)=0.25*(1+y);
                dhdr(4)=-0.25*(1+y);
                dhds(1)=-0.25*(1-x);
                dhds(2)=-0.25*(1+x);
                dhds(3)=0.25*(1+x);
                dhds(4)=0.25*(1-x);

        jacob=zeros(2,2);
        for i=1:No_nel                             % compute Jacobian
            jacob(1,1)=jacob(1,1)+dhdr(i)*xcoord(i);
            jacob(1,2)=jacob(1,2)+dhdr(i)*ycoord(i);
            jacob(2,1)=jacob(2,1)+dhds(i)*xcoord(i);
            jacob(2,2)=jacob(2,2)+dhds(i)*ycoord(i);
        end
        detjacob=det(jacob);          % determinant of Jacobian
        invjacob=inv(jacob);          % inverse of Jacobian matrix
         for i=1:No_nel
            dhdx(i)=invjacob(1,1)*dhdr(i)+invjacob(1,2)*dhds(i);
            dhdy(i)=invjacob(2,1)*dhdr(i)+invjacob(2,2)*dhds(i);
         end
        for i=1:No_nel
            i1=(i-1)*2+1;
            i2=i1+1;
            kinmtx(1,i1)=dhdx(i);
            kinmtx(2,i2)=dhdy(i);
            kinmtx(3,i1)=dhdy(i);
            kinmtx(3,i2)=dhdx(i);
        end

           k=k+kinmtx'*matmtrx*kinmtx*wtx*wty*detjacob;
         end
      end
%-------------------------------------------------------------------------
% element mass matrix
%-------------------------------------------------------------------------
area=0.5*((xcoord(1)*ycoord(2)+xcoord(2)*ycoord(3)+xcoord(3)*ycoord(1)-
 ...
   xcoord(1)*ycoord(3)-xcoord(2)*ycoord(1)-xcoord(3)*ycoord(2))+ ...
   (xcoord(1)*ycoord(3)+xcoord(3)*ycoord(4)+xcoord(4)*ycoord(1)-  ...
   xcoord(1)*ycoord(4)-xcoord(3)*ycoord(1)-xcoord(4)*ycoord(3)));
if Opt_mass==1                                 % consistent mass matrix
   m=Prop(3)*thickness*area/36*[4 0 2 0 1 0 2 0; ...
      0 4 0 2 0 1 0 2; ...
      2 0 4 0 2 0 1 0;  ;...
      0 2 0 4 0 2 0 1;  ...
```

```
        1 0 2 0 4 0 2 0;  ...
        0 1 0 2 0 4 0 2;  ...
        2 0 1 0 2 0 4 0;  ...
        0 2 0 1 0 2 0 4];
 else                                         % lumped mass matrix
    m=Prop(3)*thickness*area/4*eye(8);
 end
 %          ------------------------------------------------------
 % output element stiffness matrix and element mass matrix
 %----------------------------------------------------------------
    varargout{1}=k;
    varargout{2}=m;
  end
end
%-----------------------------------------------------------
%  The end
%-----------------------------------------------------------
```

第7章　板　结　构

在工程实际中有大量的板壳结构，这类结构的特点是其在一个方向上的尺度比在其他两个方向上的尺度相对来说小很多. 针对这个特点，可以将此类结构简化为二维问题，这样可以在保持足够精度的同时，有效降低该类问题的求解难度. 一般的板壳结构在经过简化和空间离散化之后，在局部都可以认为是薄板弯曲问题. 因此板结构单元一般都是基于薄板弯曲理论建立起来的.

7.1　经典薄板弯曲理论

如第1章所述，为了完成式(1-5)中的积分，首先需要将结构在空间上离散化，然后给出每一单元内的位移近似表达式. 在寻找单元内位移近似表达式的时候，一般都根据单元的受力特点和其本身的几何特性，对单元内的位移场做出一定的假设来将问题进行简化，这样能够较容易地得到合理的位移近似表达式. 因此下面在介绍建立板单元之前首先介绍经典薄板弯曲理论.

在研究工程中常见的薄板弯曲问题时，可以根据其几何尺寸特点，采取以下假定：

(1) 可以忽略板厚度方向的正应力，并假定薄板的厚度没有变化.

(2) 薄板的法线，在产生弯曲后，仍然保持为薄板弹性曲面的法线.

(3) 薄板中面上的各点，没有平行于中面的位移.

基于以上假设，可以推知薄板内的各应力分量和应变分量都可以用板的挠度 w 来表示. 如图7.1所示，取板的中面为 xy 平面，z 轴垂直于中面.

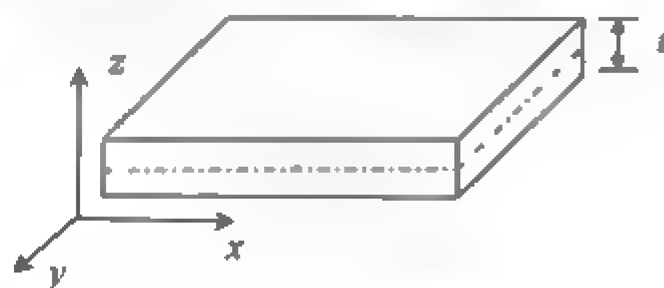


图7.1　薄板

由假设(1)可知 $\varepsilon_z = \dfrac{\partial w}{\partial z} = 0$，这样在单元中有 $w = w(x, y)$，即挠度 w 仅是 x 和 y 的函数，与 z 无关.

由假设(2)可知板内不存在剪应变，即 $\gamma_{yz} = \dfrac{\partial v}{\partial z} + \dfrac{\partial w}{\partial y} = 0, \gamma_{zx} = \dfrac{\partial w}{\partial x} + \dfrac{\partial u}{\partial z} = 0$. 可以从中推得

$$\frac{\partial v}{\partial z} = -\frac{\partial w}{\partial y}, \frac{\partial u}{\partial z} = -\frac{\partial w}{\partial x} \tag{7-1}$$

对式(7-1)进行积分，并注意到由于 w 与 z 无关，所以 $\frac{\partial w}{\partial x}$ 和 $\frac{\partial w}{\partial y}$ 也与 z 无关，因此可得

$$v=-z\frac{\partial w}{\partial y}+f_1(x,y),u=-z\frac{\partial w}{\partial x}+f_2(x,y) \tag{7-2}$$

式中，$f_1(x,y)$ 和 $f_2(x,y)$ 为关于 x 和 y 的任意函数.

由假设(3)可知 $u_{z=0}=v_{z=0}=0$，因此由式(7-2)可知 $f_1(x,y)=f_2(x,y)=0$，这样式(7-2)可以简化为

$$v=-z\frac{\partial w}{\partial y},u=-z\frac{\partial w}{\partial x} \tag{7-3}$$

综合以上推论可知，在受弯薄板中只有 3 个应变分量 $\varepsilon_x,\varepsilon_y$ 和 γ_{xy} 不为零

$$\begin{aligned}\varepsilon_x&=\frac{\partial u}{\partial x}=-z\frac{\partial^2 w}{\partial x^2}\\ \varepsilon_y&=\frac{\partial v}{\partial y}=-z\frac{\partial^2 v}{\partial y^2}\\ \gamma_{xy}&=\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}=-2z\frac{\partial^2 w}{\partial xy}\end{aligned} \tag{7-4}$$

在式(7-4)中仍含有 z，可以通过引入广义应变和广义应力的概念来得到更简洁的形式.对于各向同性材料，在不考虑 σ_z 的平面情况下有如下应力应变关系

$$\begin{aligned}\sigma_x&=\frac{E}{1-\mu^2}(\varepsilon_x+\mu\varepsilon_y)\\ \sigma_y&=\frac{E}{1-\mu^2}(\mu\varepsilon_x+\varepsilon_y)\\ \tau_{xy}&=\frac{E}{2(1+\mu)}\gamma_{xy}\end{aligned} \tag{7-5}$$

将式(7-4)代入式(7-5)得

$$\begin{aligned}\sigma_x&=-\frac{E}{1-\mu^2}z(\frac{\partial^2 w}{\partial x^2}+\mu\frac{\partial^2 w}{\partial y^2})\\ \sigma_y&=-\frac{E}{1-\mu^2}z(\mu\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2})\\ \tau_{xy}&=-\frac{E}{2(1+\mu)}z\frac{\partial^2 w}{\partial x\partial y}\end{aligned} \tag{7-6}$$

上述应力将在板内分别合成为薄板内力

$$\begin{aligned}m_x&=\int_{-t/z}^{t/z}z\sigma_x\mathrm{d}z=-\frac{Et^3}{12(1-\mu^2)}\left(\frac{\partial^2 w}{\partial x^2}+\mu\frac{\partial^2 w}{\partial y^2}\right)\\ m_y&=\int_{-t/z}^{t/z}z\sigma_y\mathrm{d}z=-\frac{Et^3}{12(1-\mu^2)}\left(\mu\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}\right)\\ m_{xy}&=m_{yx}=\int_{-t/z}^{t/z}z\sigma_{xy}\mathrm{d}z=-\frac{Et^3}{12(1+\mu)}\cdot\frac{\partial^2 w}{\partial x\partial y}\end{aligned} \tag{7-7}$$

将式(7-7)写成矩阵形式为

$$\boldsymbol{m}=\begin{Bmatrix} m_x \\ m_y \\ m_{xy} \end{Bmatrix}=\frac{Et^3}{12(1-\mu^2)}\begin{Bmatrix} -\dfrac{\partial^2 w}{\partial x^2}-\mu\dfrac{\partial^2 w}{\partial y^2} \\ -\mu\dfrac{\partial^2 w}{\partial x^2}-\dfrac{\partial^2 w}{\partial y^2} \\ -(1-\mu)\dfrac{\partial^2 w}{\partial x\partial y} \end{Bmatrix}=\frac{Et^3}{12(1-\mu^2)}\begin{bmatrix} 1 & \mu & 0 \\ \mu & 1 & 0 \\ 0 & 0 & \dfrac{1-\mu}{2} \end{bmatrix}\begin{Bmatrix} -\dfrac{\partial^2 w}{\partial x^2} \\ -\dfrac{\partial^2 w}{\partial y^2} \\ -2\dfrac{\partial^2 w}{\partial x\partial y} \end{Bmatrix} \tag{7-8}$$

式(7-8)可以简写为

$$\boldsymbol{m}=\boldsymbol{D\kappa} \tag{7-9}$$

式(7-9)即为受弯薄板中的广义应力应变关系.

其中

$$\boldsymbol{\kappa}=\begin{Bmatrix} -\dfrac{\partial^2 w}{\partial x^2} \\ -\dfrac{\partial^2 w}{\partial y^2} \\ -2\dfrac{\partial^2 w}{\partial x\partial y} \end{Bmatrix}=\boldsymbol{L}w \tag{7-10}$$

$$\boldsymbol{D}=\frac{Et^3}{12(1-\mu^2)}\begin{bmatrix} 1 & \mu & 0 \\ \mu & 1 & 0 \\ 0 & 0 & \dfrac{1-\mu}{2} \end{bmatrix}=D_0\begin{bmatrix} 1 & \mu & 0 \\ \mu & 1 & 0 \\ 0 & 0 & \dfrac{1-\mu}{2} \end{bmatrix} \tag{7-11}$$

式中，$D_0=\dfrac{Et^3}{12(1-\mu^2)}$.

7.2　经典板弯曲元

在基于经典薄板弯曲理论的单元中，每个节点有3个自由度：挠度w、法线绕x轴的转动θ_x和绕y轴的转动θ_y. 即

$$\{a_i\}=\begin{Bmatrix} w_i \\ \varphi_{xi} \\ \varphi_{yi} \end{Bmatrix}=\left\{w_i,\left(\frac{\partial w}{\partial y}\right)_i,-\left(\frac{\partial w}{\partial x}\right)_i\right\}^{\mathrm{T}} \tag{7-12}$$

单元节点位移向量为

$$\boldsymbol{a}^e=\{a_1,a_2,\cdots,a_n\}^{\mathrm{T}} \tag{7-13}$$

式中，n是单元节点的数目.

在上节已经给出了此类问题下广义应变和应力的关系，现在又确定了单元位移向量，此时，为完成根据相关定义求出单元刚度矩阵和质量矩阵等的工作，只需要再确定形函数

即可. 依据所选单元节点数目选择形函数的问题可以参阅有关的有限元理论著作，此处不再详述. 下面给出计算四节点矩形单元和三节点三角形单元刚度矩阵和质量矩阵的例子，读者可以依此思路根据有关的有限元理论著作中给出的各种节点数目情况下的形函数求得其相应的单元刚度矩阵和质量矩阵.

7.2.1 四节点矩形单元

1. 单元局部坐标系和形函数

图 7.2 所示为四节点矩形单元.

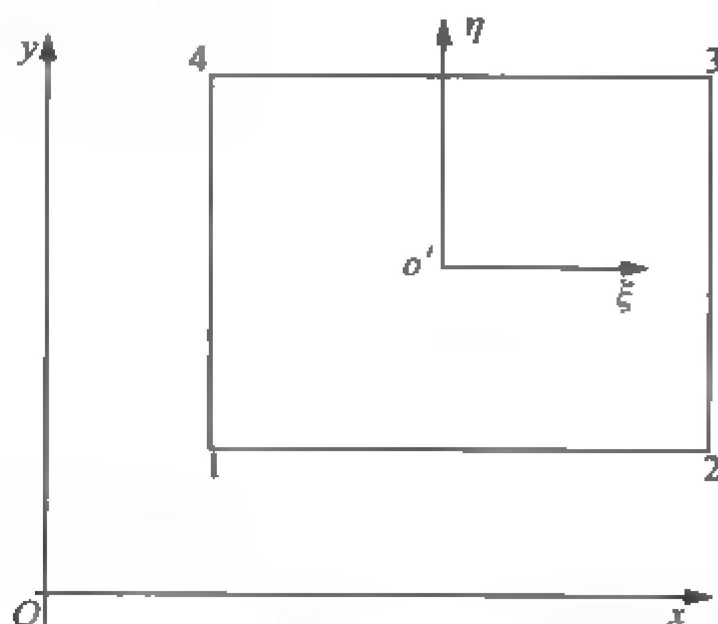


图 7.2 四节点矩形单元

如图 7.2 所示的等厚度矩形单元共有 4 个节点, 单元宽度为 a, 高度为 b. 在单元的中心 (x_0, y_0) 建立局部坐标系

$$\left.\begin{aligned} x &= x_0 + a\xi \\ y &= y_0 + b\eta \end{aligned}\right\} \tag{7-14}$$

其中

$$\left.\begin{aligned} a &= \frac{x_2 - x_1}{2} \\ b &= \frac{y_3 - x_1}{2} \end{aligned}\right\}, \quad \left.\begin{aligned} x_0 &= \frac{x_2 + x_1}{2} \\ y_0 &= \frac{y_3 + x_1}{2} \end{aligned}\right\} \tag{7-15}$$

则单元内任一点的挠度可用形函数 $N_i(x, y)$ 和单元节点位移 $\{a_i\}$ 表示为

$$w = \sum_{i=1}^{4}(N_i w_i + N_{ix}\varphi_{xi} + N_{iy}\varphi_{yi}) = \sum_{j=1}^{4}[N_j]\{a_j\} \tag{7-16}$$

其中

$$\left[N_j\right] = [N_j, N_{jx}, N_{jy}] \tag{7-17}$$

对于此类型单元，可以选用下面的形函数

$$\left.\begin{aligned}N_i&=\frac{1}{8}(1+\xi\xi_i)(1+\eta\eta_i)(2+\xi\xi_i+\eta\eta_i-\xi^2-\eta^2)\\N_{ix}&=-\frac{1}{8}b\eta_i(1+\xi\xi_i)(1+\eta\eta_i)(1-\eta^2)\\N_{iy}&=\frac{1}{8}a\xi_i(1+\xi\xi_i)(1+\eta\eta_i)(1-\xi^2)\end{aligned}\right\}\tag{7-18}$$

$$i=1,2,3,4$$

可以将式(7-16)写成矩阵形式为

$$w=Na^e\tag{7-19}$$

采取式(7-18)中形函数的时候，相邻单元之间法向导数的连续性要求一般不能得到满足，因此此类单元成为非协调单元. 不过实际计算结果表明，当单元网格划分逐步密集的时候，采取此类单元的计算结果还是能够收敛于正确解答的.

2. 单元刚度矩阵

将式(7-19)代入式(7-10)中可得

$$\kappa=Lw=LNa^e=Ba^e\tag{7-20}$$

其中 B 是3×12阶矩阵

$$B=-\begin{bmatrix}\dfrac{\partial^2N_1}{\partial x^2}&\dfrac{\partial^2N_{1x}}{\partial x^2}&\dfrac{\partial^2N_{1y}}{\partial x^2}&\cdots&\dfrac{\partial^2N_{4y}}{\partial x^2}\\\dfrac{\partial^2N_1}{\partial y^2}&\dfrac{\partial^2N_{1x}}{\partial y^2}&\dfrac{\partial^2N_{1y}}{\partial y^2}&\cdots&\dfrac{\partial^2N_{4y}}{\partial y^2}\\2\dfrac{\partial^2N_1}{\partial x\partial y}&2\dfrac{\partial^2N_{1x}}{\partial x\partial y}&2\dfrac{\partial^2N_{1y}}{\partial x\partial y}&\cdots&2\dfrac{\partial^2N_{4y}}{\partial x\partial y}\end{bmatrix}\tag{7-21}$$

此时便可根据单元刚度矩阵的定义 $K^e=\int_{V^e}B^TDB\mathrm{d}V$，求出单元刚度矩阵. 对于此处的等厚度单元，此积分结果可以显式给出. 因为该矩阵比较冗长，这里不再写出.

3. 单元质量矩阵

此种单元的集中质量矩阵是 12 阶的对角矩阵，由于可以略去转动自由度上的质量，实际上在矩阵中只有4个非零元素. 可以用以下MATLAB代码给出该矩阵：

```
m(1:12,1:12)=0;
w=a*b*t* density;
m(1,1)=w;
m(4,4)=w;
m(7,7)=w;
m(10,10)=w;
```

协调质量矩阵需要根据 $M^e=\int_{V^e}\rho N^TN\mathrm{d}V$ 求出. 这一工作可以使用 MATLAB 的符号运算功能来完成：

```
w=a*b*t*density;
syms kx yt kxi yti real;
ni=1/8*(1+kx*kxi)*(1+yt*yti)*(2+kx*kxi+yt*yti-kx^2-yt^2);
nix=-1/8*b*yti*(1+kx*kxi)*(1+yt*yti)*(1-yt^2);
niy=1/8*a*kxi*(1+kx*kxi)*(1+yt*yti)*(1-kx^2);
n(1)=subs(ni,{kxi,yti},{-1,-1});
n(2)=subs(nix,{kxi,yti},{-1,-1});
n(3)=subs(niy,{kxi,yti},{-1,-1});
n(4)=subs(ni,{kxi,yti},{1,-1});
n(5)=subs(nix,{kxi,yti},{1,-1});
n(6)=subs(niy,{kxi,yti},{1,-1});
n(7)=subs(ni,{kxi,yti},{1,1});
n(8)=subs(nix,{kxi,yti},{1,1});
n(9)=subs(niy,{kxi,yti},{1,1});
n(10)=subs(ni,{kxi,yti},{-1,1});
n(11)=subs(nix,{kxi,yti},{-1,1});
n(12)=subs(niy,{kxi,yti},{-1,1});
temp=n'*n;
m1=int(temp,kx,-1,1);
m=int(m1,yt,-1,1);
m=m*w;
m=double(m);
```

在上述代码中，因为在积分中是按照长宽厚都为1的单元来计算的，所以在代码中通过对结果乘以a*b*t以使其符合一般几何尺寸单元.

4. 单元载荷向量

当此类单元上作用有分布载荷 q 时，单元载荷向量可以按照下式计算

$$P^e = \iint_{\Omega} N^{\mathrm{T}} q \mathrm{d}x\mathrm{d}y \tag{7-22}$$

当 q 为常数时，上述积分结果为

$$P^e = 4qab\begin{bmatrix} \frac{1}{4} & \frac{b}{12} & \frac{-a}{12} & \frac{1}{4} & \frac{b}{12} & \frac{a}{12} & \frac{1}{4} & \frac{-b}{12} & \frac{a}{12} & \frac{1}{4} & \frac{-b}{12} & \frac{-a}{12} \end{bmatrix}^{\mathrm{T}} \tag{7-23}$$

5. 坐标变换问题

在第3章中指出，对于不同几何外形的同类单元，可以先推导出该类单元在标准尺寸下的各矩阵，然后再通过坐标变换得到不同外形单元的相应矩阵. 这样做要比直接推导各单元矩阵方便. 不过在应用本节单元的时候要注意：由于一般四边形不能满足常应变准则，即不能通过分片试验，所以收敛性差，不适合在实际问题中采用，只有平行四边形单元无此问题. 原因是上述推导过程中使用的标准矩形和平行四边形之间坐标变换的雅克比行列式为常数(参考第4章等参单元). 所以在使用此矩形单元的时候，要注意需要将单元划分为平行四边形.

7.2.2　三节点三角形单元

相比四边形单元，三角形单元更能够适合实际工程中结构复杂的边界形状，所以三角形单元在工程中得到较多的应用.

在三角形单元中，一般使用面积坐标来构成形函数.

1. 面积坐标

三角形中的任意一点可以和三角形的 3 个顶点组合，将三角形划分为 3 个子三角形. 可以用子三角形中属于原三角形的边所对的原三角形顶点编号来对子三角形的面积编号. 如图 7.3 所示，子三角形 *Pjm* 的面积编号为 A_i. 类似的，子三角形 *Pmi* 和子三角形 *Pij* 的面积分别为 A_j 和 A_m.

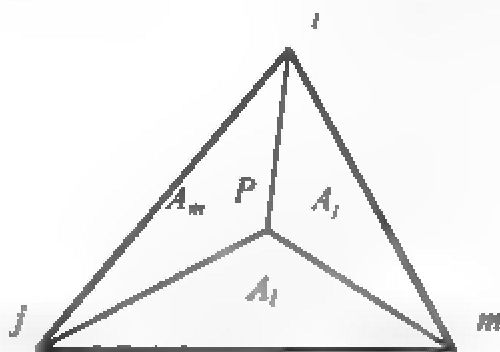


图 7.3　三角形单元

定义

$$
\begin{aligned}
L_i &= A_i / A \\
L_j &= A_j / A \\
L_m &= A_m / A
\end{aligned}
\tag{7-24}
$$

则此时 *P* 的位置可以通过以上 3 个比值来确定，此坐标称为面积坐标.

因为 3 个子三角形的面积总和应该等于原三角形面积，所以有

$$
L_i + L_j + L_m = 1 \tag{7-25}
$$

即 3 个比值(面积坐标)并不完全相互独立，而是只有两个独立量，通过任意两个可以求得第 3 个面积坐标值.

在面积坐标中，三角形内与节点 *i* 的对边 *jm* 平行的直线上的各点具有相同的 L_i 坐标. 3 个顶点的坐标分别为 *i*(1,0,0)，*j*(0,1,0)，*m*(0,0,1). 3 条边的也可以方便地表示为：*jm* 边 L_i=0；*mi* 边 L_j=0；*ij* 边 L_m=0.

2. 形函数

在使用面积坐标的情况下，对于本节三角形单元，可以使用如下形函数

$$\boldsymbol{N}_1=\begin{Bmatrix}N_1\\N_{x1}\\N_{y1}\end{Bmatrix}=\begin{Bmatrix}L_1+L_1^2L_2+L_1^2L_3-L_1L_2^2-L_1L_3^2\\b_3\left(L_3L_1^2+\frac{1}{2}L_1L_2L_3\right)-b_3\left(L_1^2L_2+\frac{1}{2}L_1L_2L_3\right)\\c_3\left(L_3L_1^2+\frac{1}{2}L_1L_2L_3\right)-c_3\left(L_1^2L_2+\frac{1}{2}L_1L_2L_3\right)\end{Bmatrix} \tag{7-26}$$

其中b_i和c_i为：

$$\left.\begin{aligned}b_i&=y_j-y_m\\c_i&=-x_j-x_m\end{aligned}\right\} \tag{7-27}$$

因为此单元有3个节点，所以还有$\boldsymbol{N}_2$和$\boldsymbol{N}_3$，可以通过对式(7-27)轮换下标来得到。

3. 计算单元刚度矩阵和质量矩阵

在确定了形函数之后，可以按照各种单元通用的标准过程完成积分得到各单元矩阵。在积分中可以使用下式来简化积分过程

$$\iint_A L_i^a L_j^b L_m^c \mathrm{d}x\mathrm{d}y=\frac{a!b!c!}{(a+b+c+2)!}2A \tag{7-28}$$

例如通过式(7-28)可以得知

$$\iint_A L_i^2 \mathrm{d}x\mathrm{d}y=\frac{2!0!0!}{(2+0+0+2)!}2A=\frac{A}{6} \tag{7-29}$$

7.3 剪切变形板元

根据考虑剪切变形的Mindlin板理论，垂直于板中面的平面在板变形后将不再垂直于中面，所以需要同时考虑板的剪切能量和弯曲能量

$$U=\frac{1}{2}\int_V \boldsymbol{\sigma}_b^{\mathrm{T}}\boldsymbol{\varepsilon}_b \mathrm{d}V+\frac{k}{2}\int_V \boldsymbol{\sigma}_S^{\mathrm{T}}\boldsymbol{\varepsilon}_S \mathrm{d}V \tag{7-30}$$

其中弯曲应力和应变为

$$\boldsymbol{\sigma}_b=\{\sigma_x \quad \sigma_y \quad \tau_{xy}\}^{\mathrm{T}} \tag{7-31}$$

$$\boldsymbol{\varepsilon}_b=\{\varepsilon_x \quad \varepsilon_y \quad \gamma_{xy}\}^{\mathrm{T}} \tag{7-32}$$

剪切应力和应变为

$$\boldsymbol{\sigma}_s=\{\tau_{xz} \quad \tau_{yz}\}^{\mathrm{T}} \tag{7-33}$$

$$\boldsymbol{\varepsilon}_s=\{\gamma_{xz} \quad \gamma_{yz}\}^{\mathrm{T}} \tag{7-34}$$

k是考虑实际的剪切应变沿厚度方向非均匀分布而引入的校正系数，为$5/6$。

将弯曲和剪切应力应变关系带入式(7-30)，可得

$$U=\frac{1}{2}\int_V \boldsymbol{\varepsilon}_b^{\mathrm{T}}\boldsymbol{D}_b\boldsymbol{\varepsilon}_b \mathrm{d}V+\frac{k}{2}\int_V \boldsymbol{\varepsilon}_s^{\mathrm{T}}\boldsymbol{D}_s\boldsymbol{\varepsilon}_s \mathrm{d}V \tag{7-35}$$

其中

$$\boldsymbol{D}_b=\frac{E}{1-\nu^2}\begin{bmatrix}1 & \nu & 0\\ \nu & 1 & 0\\ 0 & 0 & \dfrac{1-\nu}{2}\end{bmatrix} \tag{7-36}$$

$$\boldsymbol{D}_s=\begin{bmatrix}G & 0\\ 0 & G\end{bmatrix} \tag{7-37}$$

板内的位移可以表示为

$$u=-z\theta_x(x,y) \tag{7-38}$$

$$v=-z\theta_y(x,y) \tag{7-39}$$

$$w=w(x,y) \tag{7-40}$$

式中，θ_x 和 θ_y 为板中面绕 y 轴和 x 轴的转角. 假设板中面没有板内位移，则对剪切变形板，有

$$\theta_x=\frac{\partial w}{\partial x}-\gamma_{xz} \tag{7-41}$$

$$\theta_y=\frac{\partial w}{\partial y}-\gamma_{yz} \tag{7-42}$$

式中，γ 为由横向剪切变形造成的角度.

由于位能表达式中的 w,θ_x 和 θ_y 是各自独立的，所以对其进行的插值也可以各自独立进行. 这样其插值函数只要求 C_0 的连续性. 应用在等参单元一节中介绍的形函数，w,θ_x 和 θ_y 可以分别插值表示为

$$w=\sum_{i=1}^{n}H_i(\xi,\eta)wi \tag{7-43}$$

$$\theta_x=\sum_{i=1}^{n}H_i(\xi,\eta)(\theta_x)_i \tag{7-44}$$

$$\theta_y=\sum_{i=1}^{n}H_i(\xi,\eta)(\theta_y)_i \tag{7-45}$$

式中，n 为单元节点个数.

下面以双线性等参形函数为例来说明进一步的计算过程. 使用高阶形函数也可以用类似的推导过程完成计算. 弯曲和剪切应变可以从位移中计算出

$$\boldsymbol{\varepsilon}_b=-z\boldsymbol{B}_b\boldsymbol{d}^e \tag{7-46}$$

$$\boldsymbol{\varepsilon}_s=\boldsymbol{B}_s\boldsymbol{d}^e \tag{7-47}$$

其中

$$\boldsymbol{B}_b=\begin{bmatrix}\dfrac{\partial H_1}{\partial x} & 0 & 0 & \dfrac{\partial H_2}{\partial x} & 0 & 0 & \dfrac{\partial H_3}{\partial x} & 0 & 0 & \dfrac{\partial H_4}{\partial x} & 0 & 0\\ 0 & \dfrac{\partial H_1}{\partial y} & 0 & 0 & \dfrac{\partial H_2}{\partial y} & 0 & 0 & \dfrac{\partial H_3}{\partial y} & 0 & 0 & \dfrac{\partial H_4}{\partial y} & 0\\ \dfrac{\partial H_1}{\partial y} & \dfrac{\partial H_1}{\partial x} & 0 & \dfrac{\partial H_2}{\partial y} & \dfrac{\partial H_2}{\partial x} & 0 & \dfrac{\partial H_3}{\partial y} & \dfrac{\partial H_3}{\partial x} & 0 & \dfrac{\partial H_4}{\partial y} & \dfrac{\partial H_4}{\partial x} & 0\end{bmatrix} \tag{7-48}$$

$$
\boldsymbol{B}_s=\begin{bmatrix} -H_1 & 0 & \frac{\partial H_1}{\partial x} & -H_2 & 0 & \frac{\partial H_2}{\partial x} & -H_3 & 0 & \frac{\partial H_3}{\partial x} & -H_4 & 0 & \frac{\partial H_4}{\partial x} \\ 0 & -H_1 & \frac{\partial H_1}{\partial y} & 0 & -H_2 & \frac{\partial H_2}{\partial y} & 0 & -H_3 & \frac{\partial H_3}{\partial y} & 0 & -H_4 & \frac{\partial H_4}{\partial y} \end{bmatrix} \tag{7-49}
$$

$$
\boldsymbol{d}^e=\left\{(\theta_x)_1 \quad (\theta_y)_1 \quad \omega_1 \quad (\theta_x)_2 \quad (\theta_y)_2 \quad \omega_2 \quad (\theta_x)_3 \quad (\theta_y)_3 \quad \omega_3 \quad (\theta_x)_4 \quad (\theta_y)_4 \quad \omega_4\right\}^{\mathrm{T}} \tag{7-50}
$$

将式(7-46)和式(7-47)带入式(7-45)可得

$$
U=\frac{1}{2}(\boldsymbol{d}^e)^{\mathrm{T}}\int_{\Omega^e}\int_z \boldsymbol{B}_b^{\mathrm{T}}\boldsymbol{D}_b\boldsymbol{B}_b\mathrm{d}z\mathrm{d}\Omega(\boldsymbol{d}^e)+\frac{k}{2}(\boldsymbol{d}^e)^{\mathrm{T}}\int_{\Omega^e}\int_z \boldsymbol{B}_s^{\mathrm{T}}\boldsymbol{D}_s\boldsymbol{B}_s\mathrm{d}z\mathrm{d}\Omega\left\{d^e\right\} \tag{7-51}
$$

由此可得弯曲板的单元刚度矩阵

$$
\boldsymbol{K}^e=\frac{h^3}{12}\int_{\Omega^e}\boldsymbol{B}_b^{\mathrm{T}}\boldsymbol{D}_b\mathrm{d}\Omega+\kappa h\int_{\Omega^e}\boldsymbol{B}_s^{\mathrm{T}}\boldsymbol{D}_s\boldsymbol{B}_s\mathrm{d}\Omega \tag{7-52}
$$

式中，h 为板的厚度.

当板的厚度相对于其边长很小的时候，剪切能量将远大于弯曲能量，这种现象称为剪切锁死. 为避免此问题，一般可以选用选择积分或者缩减积分的方法. 研究结果还表明，在避免剪切锁死方面，Lagrange 单元通常具有较好的性能.

7.4 具有位移自由度的板元

在此类单元中，认为单元内位移中的 u 和 v 不但在 xy 平面内变化(此处设坐标系中 xy 平面与板的中面平行)，而且也有沿板的厚度方向(z 轴方向)的变化. 所以是 x,y 和 z 的函数

$$
u=u(x,y,z) \tag{7-53}
$$

$$
v=v(x,y,z) \tag{7-54}
$$

而 w 则没有沿板的厚度方向的变化，可以表示为

$$
w=w(x,y) \tag{7-55}
$$

所以在使用插值函数和各节点位移来表示板内任意一点的位移的时候，使用两个插值函数 $N_i(x,y)$ 和 $H_j(z)$. 其中 $N_i(x,y)$ 用以完成在 xy 平面内变化的插值，而 $H_j(z)$ 用以完成沿 z 轴变化的插值. 单元的节点位移也使用两个下标来进行编号. 如：u_{ij}. 其中 i 是在 xy 平面内的编号，而 j 是在 z 方向的编号. 对于 w_{ij}，因为具有相同 i 的各点的 w 位移都相同，所以可以略去第 2 个下标，直接记为 w_i.

这样板内任意一点的位移可以表示为(在此处使用了局部坐标系 $\xi\eta\varsigma$)

$$
u=\sum_{i=1}^{N_1}\sum_{j=1}^{N_2}N_i(\xi,\eta)H_j(\zeta)u_{ij} \tag{7-56}
$$

$$
v=\sum_{i=1}^{N_1}\sum_{j=1}^{N_2}N_i(\xi,\eta)H_j(\zeta)v_{ij} \tag{7-57}
$$

$$
w=\sum_{i=1}^{N_1}N_i(\xi,\eta)w_i \tag{7-58}
$$

式中，N_1 和 N_2 为在 $\xi\eta$ 平面(xy 平面)内节点的数目和沿 ζ 轴(z 轴)节点的数目. 例如对于选取立方体全部顶点作为节点的板单元来说，$N_1=4$，$N_2=2$(即单元在矩形上下表面上各有4个节点).

弯曲应变 $\boldsymbol{\varepsilon}_b$ 和横向剪切应变 $\boldsymbol{\varepsilon}_s$ 分别可以用位移表示为

$$\boldsymbol{\varepsilon}_b=\begin{Bmatrix}\varepsilon_x\\ \varepsilon_y\\ \gamma_{xy}\end{Bmatrix}=\begin{bmatrix}\dfrac{\partial}{\partial x} & 0 & 0\\ 0 & \dfrac{\partial}{\partial y} & 0\\ \dfrac{\partial}{\partial y} & \dfrac{\partial}{\partial x} & 0\end{bmatrix}\begin{Bmatrix}u\\ v\\ w\end{Bmatrix} \tag{7-59}$$

$$\boldsymbol{\varepsilon}_s=\begin{Bmatrix}\gamma_{yz}\\ \gamma_{zx}\end{Bmatrix}=\begin{bmatrix}\dfrac{\partial}{\partial z} & 0 & \dfrac{\partial}{\partial x}\\ 0 & \dfrac{\partial}{\partial z} & \dfrac{\partial}{\partial y}\end{bmatrix}\begin{Bmatrix}u\\ v\\ w\end{Bmatrix} \tag{7-60}$$

在此类单元计算过程中，垂直于板中面的应变 $\boldsymbol{\varepsilon}_z$ 可以忽略不计.

对 $N_1=4, N_2=2$ 的选取立方体全部顶点作为节点的单元，将式(7-56)～式(7-58)代入式(7-59)和式(7-60)，可得

$$\boldsymbol{\varepsilon}_b=\boldsymbol{B}_b\boldsymbol{d}^e \tag{7-61}$$

$$\boldsymbol{\varepsilon}_s=\boldsymbol{B}_s\boldsymbol{d}^e \tag{7-62}$$

其中

$$\boldsymbol{B}_b=\begin{bmatrix}\boldsymbol{B}_{b1} & \boldsymbol{B}_{b2} & \boldsymbol{B}_{b3} & \boldsymbol{B}_{b4}\end{bmatrix} \tag{7-63}$$

$$\boldsymbol{B}_{bi}=\begin{bmatrix}H_1\dfrac{\partial N_i}{\partial x} & 0 & H_2\dfrac{\partial N_i}{\partial x} & 0 & 0\\ 0 & H_1\dfrac{\partial N_i}{\partial y} & 0 & H_2\dfrac{\partial N_i}{\partial y} & 0\\ H_1\dfrac{\partial N_i}{\partial y} & H_1\dfrac{\partial N_i}{\partial x} & H_2\dfrac{\partial N_i}{\partial y} & H_2\dfrac{\partial N_i}{\partial x} & 0\end{bmatrix} \tag{7-64}$$

$$\boldsymbol{d}^e=\begin{Bmatrix}d_1^e & d_2^e & d_3^e & d_4^e\end{Bmatrix}^{\mathrm{T}} \tag{7-65}$$

$$\boldsymbol{d}_i^e=\begin{Bmatrix}u_{i1} & v_{i1} & u_{i2} & v_{i2} & w_i\end{Bmatrix} \tag{7-66}$$

$$\boldsymbol{B}_s=\begin{bmatrix}\boldsymbol{B}_{s1} & \boldsymbol{B}_{s2} & \boldsymbol{B}_{s3} & \boldsymbol{B}_{s4}\end{bmatrix} \tag{7-67}$$

$$\boldsymbol{B}_{si}=\begin{bmatrix}N_i\dfrac{\partial H_1}{\partial z} & 0 & N_i\dfrac{\partial H_2}{\partial z} & 0 & \dfrac{\partial N_1}{\partial x}\\ 0 & N_i\dfrac{\partial H_1}{\partial z} & 0 & N_i\dfrac{\partial H_2}{\partial z} & \dfrac{\partial H_2}{\partial y}\end{bmatrix} \tag{7-68}$$

各向同性材料的本构方程是

$$\boldsymbol{\sigma}_b=\boldsymbol{D}_b\boldsymbol{\varepsilon}_b \tag{7-69}$$

其中

$$\boldsymbol{\sigma}_b=\left\{\sigma_x \quad \sigma_y \quad \tau_{xy}\right\}^{\mathrm{T}} \tag{7-70}$$

$$\boldsymbol{D}_b=\frac{E}{1-\nu^2}\begin{bmatrix}1 & \nu & 0\\ \nu & 1 & 0\\ 0 & 0 & \frac{1-\nu}{2}\end{bmatrix} \tag{7-71}$$

对于弯曲分量有

$$\boldsymbol{\sigma}_s=\boldsymbol{D}_s\boldsymbol{\varepsilon}_s \tag{7-72}$$

其中

$$\boldsymbol{\sigma}_s=\left\{\tau_{yz} \quad \tau_{zx}\right\}^{\mathrm{T}} \tag{7-73}$$

$$\boldsymbol{D}_s=\frac{E}{2(1+\nu)}\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} \tag{7-74}$$

系统的总势能可以表示为

$$\Pi=U-W \tag{7-75}$$

其中外力所做的功 W 为

$$W=\boldsymbol{d}^{\mathrm{T}}\boldsymbol{F} \tag{7-76}$$

在式(7-76)中 $\boldsymbol{d}$ 是系统节点位移向量，$\boldsymbol{F}$ 是系统载荷向量. 因为在此单元中没有考虑转动自由度，所以外力矩应该转换为一对分别作用在单元上下表面上的力来表示.

式(7-75)中内应变能 U 由两部分组成

$$U=U_b+U_s \tag{7-77}$$

其中弯曲应变能量为

$$U_b=\frac{1}{2}\int_{\Omega}\boldsymbol{\sigma}_b^{\mathrm{T}}\boldsymbol{\varepsilon}_b\,\mathrm{d}\Omega \tag{7-78}$$

横向剪切应变能量为

$$U_s=\frac{1}{2}\int_{\Omega}\boldsymbol{\sigma}_s^{\mathrm{T}}\boldsymbol{\varepsilon}_s\,\mathrm{d}\Omega \tag{7-79}$$

其中积分区域 Ω 为整个板结构. 实际上，在有限元中，该积分是通过对结构的空间离散化来分区域进行的. 所以上述积分可以写成各子区域(单元)积分的总和

$$U_b=\sum_e\frac{1}{2}(\boldsymbol{d}^e)^{\mathrm{T}}\int_{\Omega^e}\boldsymbol{B}_b^{\mathrm{T}}\boldsymbol{D}_b\boldsymbol{B}_b\,\mathrm{d}\Omega(\boldsymbol{d}^e) \tag{7-80}$$

$$U_s=\sum_e\frac{1}{2}(\boldsymbol{d}^e)^{\mathrm{T}}\int_{\Omega^e}\boldsymbol{B}_s^{\mathrm{T}}\boldsymbol{D}_s\boldsymbol{B}_s\,\mathrm{d}\Omega(\boldsymbol{d}^e) \tag{7-81}$$

在上面两式中 e 为结构中所有单元总数.

综合上述各式，可以得出此种单元的单元刚度矩阵为

$$\boldsymbol{K}^e=\int_{\Omega^e}\boldsymbol{B}_b^{\mathrm{T}}\boldsymbol{D}_b\boldsymbol{B}_b\,\mathrm{d}\Omega+\int_{\Omega^e}\boldsymbol{B}_s^{\mathrm{T}}\boldsymbol{D}_s\boldsymbol{B}_s\,\mathrm{d}\Omega \tag{7-82}$$

在计算薄板问题时，应格外注意在通过数值积分得到式(7-82)结果的时候，应该使用缩减积分方案来避免出现剪切锁死现象.

7.5 混合板元

经典板理论的方程为

$$M_x=-D_r\left(\frac{\partial^2 w}{\partial x^2}+\nu\frac{\partial^2 w}{\partial y^2}\right) \tag{7-83}$$

$$M_y=-D_r\left(\frac{\partial^2 w}{\partial y^2}+\nu\frac{\partial^2 w}{\partial x^2}\right) \tag{7-84}$$

$$M_{xy}=-D_r(1-\nu)\frac{\partial^2 w}{\partial x\partial y} \tag{7-85}$$

$$\frac{\partial^2 M_x}{\partial x^2}+\frac{\partial^2 M_y}{\partial x^2}+2\frac{\partial^2 M_{xy}}{\partial x\partial y}=-p \tag{7-86}$$

式中：M 为弯矩；$D_r=\dfrac{Eh^3}{12\left(1-\nu^2\right)}$ 为板的弯曲刚度；E 为材料的弹性模量；ν 为材料的泊松比；h 为板的厚度；p 为压力载荷．

直接对式(7-83)～式(7-85)使用伽辽金法求解，并不能得到对称矩阵的结果．所以首先将式(7-83)～式(7-85)转换为以下各式

$$S\left(M_x-\nu M_y\right)+\frac{\partial^2 w}{\partial x^2}=0 \tag{7-87}$$

$$S\left(M_y-\nu M_x\right)+\frac{\partial^2 w}{\partial y^2}=0 \tag{7-88}$$

$$2S(1+\nu)M_{xy}+2\frac{\partial^2 w}{\partial x\partial y}=0 \tag{7-89}$$

其中 $S=\dfrac{12}{Eh^3}$．这样可以对式(7-86)中的平衡方程使用伽辽金法求解．单个单元的方程如下

$$\begin{bmatrix} K_1 & K_2 & 0 & K_3 \\ K_2 & K_1 & 0 & K_4 \\ 0 & 0 & K_5 & K_6 \\ K_3 & K_4 & K_6 & 0 \end{bmatrix}\begin{Bmatrix} M_x \\ M_y \\ M_{xy} \\ w \end{Bmatrix}=\begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{Bmatrix} \tag{7-90}$$

其中

$$K_1=S\int_{\Omega^e} N^{\mathrm{T}} N\mathrm{d}\Omega \tag{7-91}$$

$$K_2=-\nu K_1 \tag{7-92}$$

$$K_3=-\int_{\Omega^e}\left[\frac{\partial N}{\partial x}\right]^{\mathrm{T}}\left[\frac{\partial N}{\partial x}\right]\mathrm{d}\Omega \tag{7-93}$$

$$K_4=-\int_{\Omega^e}\left[\frac{\partial N}{\partial y}\right]^{\mathrm{T}}\left[\frac{\partial N}{\partial y}\right]\mathrm{d}\Omega \tag{7-94}$$

$$K_5 = -2(1+\nu)K_1 \tag{7-95}$$

$$K_6 = -\int_{\Omega^e}\left(\left[\frac{\partial N}{\partial x}\right]^{\mathrm{T}}\left[\frac{\partial N}{\partial y}\right]+\left[\frac{\partial N}{\partial x}\right]^{\mathrm{T}}\left[\frac{\partial N}{\partial y}\right]\right)\mathrm{d}\Omega \tag{7-96}$$

$$F_1 = -\int_{\Gamma^e} N^{\mathrm{T}}\frac{\partial w}{\partial x}l_x\mathrm{d}\Gamma \tag{7-97}$$

$$F_2 = -\int_{\Gamma^e} N^{\mathrm{T}}\frac{\partial w}{\partial y}l_y\mathrm{d}\Gamma \tag{7-98}$$

$$F_3 = -\int_{\Gamma^e} N^{\mathrm{T}}\left(\frac{\partial w}{\partial y}l_x+\frac{\partial w}{\partial x}l_y\right)\mathrm{d}\Gamma \tag{7-99}$$

$$F_4 = -\int_{\Gamma^e} N^{\mathrm{T}}Q_n\mathrm{d}\Gamma + \int_{\Gamma^e} N^{\mathrm{T}}p\mathrm{d}\Omega \tag{7-100}$$

$$Q_n = Q_x l_x + Q_y l_y \tag{7-101}$$

在上面各式中：l_x 和 l_y 为单位正交向量的方向余弦值；Q 为剪切力；N 为形函数. 各种等参单元，无论是四节点还是三节点，都可以采用以上各式进行计算.

在前面各式中没有考虑横向剪切变形的影响，在计算厚板问题时，可以采用下面给出的混合板单元.

考虑了横向剪切力的板平衡方程如下

$$\frac{\partial M_x}{\partial x}+\frac{\partial M_{xy}}{\partial y}-Q_x=0 \tag{7-102}$$

$$\frac{\partial M_{xy}}{\partial x}+\frac{\partial M_y}{\partial y}-Q_y=0 \tag{7-103}$$

$$\frac{\partial Q_x}{\partial x}+\frac{\partial Q_y}{\partial y}+p=0 \tag{7-104}$$

厚板和薄板理论的主要区别在于转角和横向变形的关系. 在薄板理论中，单元节点转角与横向变形是相关的；但是在厚板理论中，认为两者无关. 因此，单元在 x,y,z 3 个方向上的位移可以表示为

$$u=-z\theta_x(x,y) \tag{7-105}$$

$$v=-z\theta_y(x,y) \tag{7-106}$$

$$w=w(x,y) \tag{7-107}$$

式中，θ_x 和 θ_y 为关于 x 和 y 轴的转角. 将以上 3 式带入运动方程并应用本构方程得到

$$S(M_x-\nu M_y)+\frac{\partial \theta_x}{\partial x}=0 \tag{7-108}$$

$$S(M_y-\nu M_x)+\frac{\partial \theta_y}{\partial y}=0 \tag{7-109}$$

$$2S(1+\nu)M_{xy}+\frac{\partial \theta_x}{\partial y}+\frac{\partial \theta_y}{\partial x}=0 \tag{7-110}$$

在以上 3 式中，如果将 θ_x 和 θ_y 分别替换成 $\frac{\partial w}{\partial x}$ 和 $\frac{\partial w}{\partial y}$，那么将得到式(7-87)～式(7-89)

然而在厚板理论中，上述两组方程间的关系将不会保持.

运用对于横向剪切分量的运动方程和本构方程，可以将剪切力用转动和变形表示如下

$$Q_x = \kappa Gh\left(-\theta_x + \frac{\partial w}{\partial x}\right) \tag{7-111}$$

$$Q_y = \kappa Gh\left(-\theta_y + \frac{\partial w}{\partial y}\right) \tag{7-112}$$

式中：κ 为剪切校正系数，其值为 5/6；G 为剪切模量；h 为板的厚度. 从以上两式可以得到转角的表达式

$$\theta_x = -\frac{Q_x}{\kappa Gh} + \frac{\partial w}{\partial x} \tag{7-113}$$

$$\theta_y = -\frac{Q_y}{\kappa Gh} + \frac{\partial w}{\partial y} \tag{7-114}$$

将以上两式带入式(7-108)～式(7-110)，消去其中的转动项得到

$$S(M_x - \nu M_y) - \frac{1}{\kappa Gh}\frac{\partial Q_x}{\partial x} + \frac{\partial^2 w}{\partial x^2} = 0 \tag{7-115}$$

$$S(M_y - \nu M_x) - \frac{1}{\kappa Gh}\frac{\partial Q_y}{\partial y} + \frac{\partial^2 w}{\partial y^2} = 0 \tag{7-116}$$

$$2S(1+\nu)M_{xy} - \frac{1}{\kappa Gh}\left(\frac{\partial Q_x}{\partial y} + \frac{\partial Q_y}{\partial x}\right) + 2\frac{\partial^2 w}{\partial x \partial y} = 0 \tag{7-117}$$

从上面 3 式可以看出剪切力项和力矩项前的系数分别是与板厚度和板厚度 3 次方成反比，这样在板的厚度趋近于零的时候，和力矩项相比，剪切力项可以忽略不计. 考虑到当板的厚度与其长度相比很小时，板的剪切变形可以忽略不计，所以前述忽略剪切力项是可以接受的.

为了消去剪切力，将式(7-102)和式(7-103)带入式(7-115)～式(7-117)，并运用式(7-104)就可以得到

$$\left(S - \frac{1}{\kappa Gh}\frac{\partial^2}{\partial x^2}\right)M_x - \nu S M_y - \frac{1}{\kappa Gh}\frac{\partial^2 M_{xy}}{\partial x \partial y} + \frac{\partial^2 w}{\partial x^2} = 0 \tag{7-118}$$

$$-\nu S M_x + \left(S - \frac{1}{\kappa Gh}\frac{\partial^2}{\partial y^2}\right)M_y - \frac{1}{\kappa Gh}\frac{\partial^2 M_{xy}}{\partial x \partial y} + \frac{\partial^2 w}{\partial y^2} = 0 \tag{7-119}$$

$$-\frac{1}{\kappa Gh}\left(\frac{\partial^2 M_x}{\partial x \partial y} + \frac{\partial^2 M_y}{\partial x \partial y}\right) + \left(2S(1+\nu) - \frac{1}{\kappa Gh}\frac{\partial^2}{\partial x^2} - \frac{1}{\kappa Gh}\frac{\partial^2}{\partial y^2}\right)M_{xy} + 2\frac{\partial^2 w}{\partial x \partial y} = 0 \tag{7-120}$$

以上 3 式加上式(7-86)，4 个方程中包含有同样的 4 个变量：M_x, M_y, M_{xy} 和 w，这和在薄板公式中的情况一样. 如果将系数中带有 $\frac{1}{\kappa Gh}$ 的项略去，上述公式就缩减成了薄板问题的方程. 实际上，当板的厚度趋向于零时，以上各项是可以忽略的. 因为剪切力相关项都是和 h 成反比，而弯曲相关项都是和 h^3 成反比.

对以上的 4 个方程应用伽辽金法，便可得到以下矩阵形式的方程

$$\begin{bmatrix} K_{11} & K_{12} & K_{13} & K_{14} \\ K_{21} & K_{22} & K_{23} & K_{24} \\ K_{31} & K_{32} & K_{33} & K_{34} \\ K_{41} & K_{42} & K_{43} & K_{44} \end{bmatrix} \begin{Bmatrix} M_x \\ M_y \\ M_{xy} \\ w \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{Bmatrix} \tag{7-121}$$

其中

$$K_{11} = S\int_{\Omega^e} N^{\mathrm{T}} N \mathrm{d}\Omega + \frac{1}{\kappa Gh}\int_{\Omega^e}\left[\frac{\partial N}{\partial x}\right]^{\mathrm{T}}\left[\frac{\partial N}{\partial x}\right]\mathrm{d}\Omega \tag{7-122}$$

$$K_{12} = -\nu S\int_{\Omega^e} N^{\mathrm{T}} N \mathrm{d}\Omega \tag{7-123}$$

$$K_{13} = \frac{1}{\kappa Gh}\int_{\Omega^e}\left[\frac{\partial N}{\partial x}\right]^{\mathrm{T}}\left[\frac{\partial N}{\partial y}\right]\mathrm{d}\Omega \tag{7-124}$$

$$K_{14} = -\int_{\Omega^e}\left[\frac{\partial N}{\partial x}\right]^{\mathrm{T}}\left[\frac{\partial N}{\partial x}\right]^{\mathrm{T}}\mathrm{d}\Omega \tag{7-125}$$

$$K_{22} = S\int_{\Omega^e} N^{\mathrm{T}} N \mathrm{d}\Omega + \frac{1}{\kappa Gh}\int_{\Omega^e}\left[\frac{\partial N}{\partial y}\right]^{\mathrm{T}}\left[\frac{\partial N}{\partial y}\right]\mathrm{d}\Omega \tag{7-126}$$

$$K_{13} = \frac{1}{\kappa Gh}\int_{\Omega^e}\left[\frac{\partial N}{\partial y}\right]^{\mathrm{T}}\left[\frac{\partial N}{\partial x}\right]\mathrm{d}\Omega \tag{7-127}$$

$$K_{24} = -\int_{\Omega^e}\left[\frac{\partial N}{\partial y}\right]^{\mathrm{T}}\left[\frac{\partial N}{\partial y}\right]^{\mathrm{T}}\mathrm{d}\Omega \tag{7-128}$$

$$K_{33} = 2(1+\nu)S\int_{\Omega^e} N^{\mathrm{T}} N \mathrm{d}\Omega + \frac{1}{\kappa Gh}\int_{\Omega^e}\left[\frac{\partial N}{\partial x}\right]^{\mathrm{T}}\left[\frac{\partial N}{\partial x}\right]\mathrm{d}\Omega + \frac{1}{\kappa Gh}\int_{\Omega^e}\left[\frac{\partial N}{\partial y}\right]^{\mathrm{T}}\left[\frac{\partial N}{\partial y}\right]\mathrm{d}\Omega \tag{7-129}$$

$$K_{34} = -\int_{\Omega^e}\left(\left[\frac{\partial N}{\partial x}\right]^{\mathrm{T}}\left[\frac{\partial N}{\partial y}\right] + \left[\frac{\partial N}{\partial x}\right]^{\mathrm{T}}\left[\frac{\partial N}{\partial y}\right]\right)\mathrm{d}\Omega \tag{7-130}$$

$$K_{44} = 0 \tag{7-131}$$

$$F_1 = -\int_{\Gamma^e} N^{\mathrm{T}}\frac{\partial w}{\partial x} l_x \mathrm{d}\Gamma + \frac{1}{\kappa Gh}\int_{\Gamma^e} N^{\mathrm{T}} V_x l_x \mathrm{d}\Gamma \tag{7-132}$$

$$F_2 = -\int_{\Gamma^e} N^{\mathrm{T}}\frac{\partial w}{\partial y} l_y \mathrm{d}\Gamma + \frac{1}{\kappa Gh}\int_{\Gamma^e} N^{\mathrm{T}} V_y l_y \mathrm{d}\Gamma \tag{7-133}$$

$$\begin{aligned} F_3 = &-\int_{\Gamma^e} N^{\mathrm{T}}\left(\frac{\partial w}{\partial y} l_x + \frac{\partial w}{\partial x} l_y\right)\mathrm{d}\Gamma + \\ &\frac{1}{\kappa Gh}\int_{\Gamma^e} N^{\mathrm{T}}\left(\frac{\partial}{\partial y}\left(M_x l_x + M_{xy} l_y\right) + \frac{\partial}{\partial x}\left(M_{xy} l_x + M_y l_y\right)\right)\mathrm{d}\Gamma \end{aligned} \tag{7-134}$$

$$F_4 = -\int_{\Gamma^e} N^{\mathrm{T}} Q_n \mathrm{d}\Gamma + \int_{\Gamma^e} N^{\mathrm{T}} p \mathrm{d}\Omega \tag{7-135}$$

7.6 杂交板元

杂交板元也是基于对平板内应力的假设，需要 C^0 阶连续性. 和其他类型单元的区别在于杂交板元是基于如下改正过的系统势能表达式进行推导的.

$$\Pi = \int_{\Omega}(-\frac{1}{2}\boldsymbol{\varepsilon}_b^{\mathrm{T}}\boldsymbol{D}_b\boldsymbol{\varepsilon}_b - \frac{1}{2}\boldsymbol{\varepsilon}_s^{\mathrm{T}}\boldsymbol{D}_s\boldsymbol{\varepsilon}_s + \boldsymbol{\varepsilon}_b^{\mathrm{T}}\boldsymbol{D}_b\boldsymbol{L}_b\boldsymbol{d} + \boldsymbol{\varepsilon}_s^{\mathrm{T}}\boldsymbol{D}_s\boldsymbol{L}_s\boldsymbol{d})\mathrm{d}\Omega - \int_{\Gamma}\boldsymbol{d}^{\mathrm{T}}\boldsymbol{p}\mathrm{d}\Gamma \tag{7-136}$$

其中

$$\boldsymbol{\varepsilon}_b = \left\{\frac{\partial\theta_x}{\partial x} \quad \frac{\partial\theta_y}{\partial y} \quad \left(\frac{\partial\theta_x}{\partial y} + \frac{\partial\theta_y}{\partial x}\right)\right\}^{\mathrm{T}} \tag{7-137}$$

$$\boldsymbol{\varepsilon}_s = \left\{\left(-\theta_x + \frac{\partial w}{\partial x}\right) \quad \left(-\theta_y + \frac{\partial w}{\partial y}\right)\right\}^{\mathrm{T}} \tag{7-138}$$

$$\boldsymbol{d} = \{\theta_x \quad \theta_y \quad w\}^{\mathrm{T}} \tag{7-139}$$

此外 $\boldsymbol{D}_b$ 是弯曲应变相关的材料性能矩阵；$\boldsymbol{D}_s$ 是横向剪切应变相关的材料性能矩阵；$\boldsymbol{L}_b$ 是弯曲应变位移算子；而 $\boldsymbol{L}_s$ 是剪切应变位移算子；$\boldsymbol{p}$ 是作用在板上的压力载荷.

通过系统势能的驻值解得到平衡方程和广义应变位移关系. 为了得到有限元模型，将广义应变位移离散化为以下形式

$$\boldsymbol{\varepsilon}_b = \boldsymbol{B}_b\boldsymbol{\alpha}_b \tag{7-140}$$

$$\boldsymbol{\varepsilon}_s = \boldsymbol{B}_s\boldsymbol{\alpha}_s \tag{7-141}$$

$$\boldsymbol{d} = \boldsymbol{N}\hat{\boldsymbol{d}} \tag{7-142}$$

其中假设各单元中的广义应变是相互独立的. 广义位移可以使用各广义节点位移 $\boldsymbol{d}$ 插值而得. $\boldsymbol{B}_b$ 和 $\boldsymbol{B}_s$ 分别是广义应变参数向量 $\boldsymbol{\alpha}_b$ 和 $\boldsymbol{\alpha}_s$ 的多项式表达式系数的矩阵形式. $\boldsymbol{N}$ 是形函数的矩阵形式.

$$\Pi = -\frac{1}{2}\boldsymbol{\alpha}_b^{\mathrm{T}}\boldsymbol{G}_b\boldsymbol{\alpha}_b - \frac{1}{2}\boldsymbol{\alpha}_s^{\mathrm{T}}\boldsymbol{G}_s\boldsymbol{\alpha}_s + \boldsymbol{\alpha}_b^{\mathrm{T}}\boldsymbol{H}_b\hat{\boldsymbol{d}} + \boldsymbol{\alpha}_s^{\mathrm{T}}\boldsymbol{H}_s\hat{\boldsymbol{d}} - \boldsymbol{d}^{\mathrm{T}}\boldsymbol{F} \tag{7-143}$$

其中

$$\boldsymbol{G}_b = \int_{\Omega^e}\boldsymbol{B}_b^{\mathrm{T}}\boldsymbol{D}_b\boldsymbol{B}_b\mathrm{d}\Omega \tag{7-144}$$

$$\boldsymbol{G}_s = \int_{\Omega^e}\boldsymbol{B}_s^{\mathrm{T}}\boldsymbol{D}_s\boldsymbol{B}_s\mathrm{d}\Omega \tag{7-145}$$

$$\boldsymbol{H}_b = \int_{\Omega^e}\boldsymbol{B}_b^{\mathrm{T}}\boldsymbol{D}_b\boldsymbol{L}_b\boldsymbol{N}\mathrm{d}\Omega \tag{7-146}$$

$$\boldsymbol{H}_s = \int_{\Omega^e}\boldsymbol{B}_s^{\mathrm{T}}\boldsymbol{D}_s\boldsymbol{L}_s\boldsymbol{N}\mathrm{d}\Omega \tag{7-147}$$

$$\boldsymbol{F} = \int_{\Gamma^e}\boldsymbol{N}^{\mathrm{T}}\boldsymbol{p}\mathrm{d}\Gamma \tag{7-148}$$

调用式(7-143)分别关于 $\boldsymbol{\alpha}_b$ 和 $\boldsymbol{\alpha}_s$ 的驻值得到

$$-\boldsymbol{G}_b\boldsymbol{\alpha}_b + \boldsymbol{H}_b\hat{\boldsymbol{d}} = 0 \tag{7-149}$$

$$-\boldsymbol{G}_s\boldsymbol{\alpha}_s + \boldsymbol{H}_s\hat{\boldsymbol{d}} = 0 \tag{7-150}$$

运用式(7-149)和式(7-150)从式(7-143)中消去 $\boldsymbol{\alpha}_b$ 和 $\boldsymbol{\alpha}_s$ 得到

$$\Pi = \frac{1}{2}\hat{\boldsymbol{d}}^{\mathrm{T}}\left(\boldsymbol{H}_b^{\mathrm{T}}\boldsymbol{G}_b^{-\mathrm{T}}\boldsymbol{H}_b + \boldsymbol{H}_s^{\mathrm{T}}\boldsymbol{G}_s^{-\mathrm{T}}\boldsymbol{H}_s\right)\hat{\boldsymbol{d}} - \hat{\boldsymbol{d}}^{\mathrm{T}}\boldsymbol{F} \tag{7-151}$$

通过式(7-151)可以最终得到如下的有限元系统方程

$$\boldsymbol{K}\hat{\boldsymbol{d}} = \boldsymbol{F} \tag{7-152}$$

其中

$$\boldsymbol{K} = \boldsymbol{H}_b^{\mathrm{T}}\boldsymbol{G}_b^{-\mathrm{T}}\boldsymbol{H}_b + \boldsymbol{H}_s^{\mathrm{T}}\boldsymbol{G}_s^{-\mathrm{T}}\boldsymbol{H}_s \tag{7-153}$$

对于双线性板单元，广义应变向量假设为

$$\boldsymbol{B}_b = \begin{bmatrix} 1 & 0 & 0 & x & 0 & 0 & y & 0 & 0 \\ 0 & 1 & 0 & 0 & x & 0 & 0 & y & 0 \\ 0 & 0 & 1 & 0 & 0 & x & 0 & 0 & y \end{bmatrix} \tag{7-154}$$

$$\boldsymbol{B}_s = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{7-155}$$

从式(7-155)可以看出，在双线性板单元中，弯曲应变是线性变化的，剪切应变是恒定不变的常数.

7.7 非轴对称超参数壳体单元

在研究壳体问题时，通过将壳体离散近似为多个局部平板的组合，即使用前面所介绍的各种板单元来进行分析，当单元数目足够多时，可以提供足够的精度. 但是如果使用本身就是曲面的壳体单元，自然可以使用更少的单元就达到同样的精度. 因此本书不再单列一章讨论壳体单元，只介绍一种非轴对称的超参数壳体单元作为补充.

由于对于厚板和厚壳，中面法线在变形后仍基本保持为直线，所以在这种单元中假定中面法线在变形后仍保持为直线，并忽略垂直于中面的正应力所引起的应变能，每一节点有 5 个自由度. 通过利用形函数进行坐标变换，壳体中面可以是任意曲面. 在这种单元计算中还考虑了横向剪应力影响，故比一般基于薄壳理论的单元准确. 因此这种单元不但可以分析厚壳，而且可以分析薄壳. 大量的研究和计算实例表明，这种单元具有计算精度高、适用范围广等优点，是目前为止最好的壳体单元之一.

7.7.1 曲面单元与映射

如图 7.4 所示，单元的局部坐标为 (ξ,η,ζ)，$\zeta=0$ 为单元的中面，$\zeta=\pm1$ 为单元的上、下表面，均为曲面. $\xi=\pm1$ 和 $\eta=\pm1$ 是由直线产生的 4 个截面. 单元节点取在中面上，共取 8 个节点，即 4 个角点和 4 个边中点.

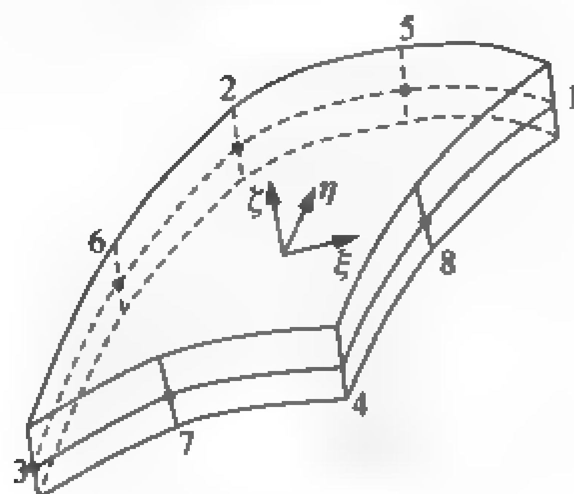


图 7.4　曲面单元

设节点 i 的直角坐标为 (x_i, y_i, z_i)，则中面 $(\zeta=0)$ 上任一点 ξ,η 的整体坐标可以表示如下

$$x=\sum N_i x_i,\quad y=\sum N_i y_i,\quad z=\sum N_i z_i \tag{7-156}$$

式中，$N_i(\xi,\eta)$ 为单元的形函数，具体表达式为

$$\left.\begin{aligned}&\text{对于角点：}N_i=\frac{1}{4}(1+\xi_i\xi)(1+\eta_i\eta)(\xi_i\xi+\eta_i\eta-1)\\&\text{对于边中点：}\xi_i=0,\quad N_i=\frac{1}{2}(1-\xi^2)(1+\eta_i\eta)\\&\qquad\qquad\quad\eta_i=0,\quad N_i=\frac{1}{2}(1+\xi_i\xi)(1-\eta^2)\end{aligned}\right\} \tag{7-157}$$

设 $\zeta=+1$ 代表单元的下表面，$\zeta=-1$ 代表单元的上表面，单元内任一点的整体坐标 (x,y,z) 可以用局部坐标 (ξ,η,ζ) 表示如下

$$\begin{Bmatrix}x\\y\\z\end{Bmatrix}=\sum N_i(\xi,\eta)\frac{(1+\zeta)}{2}\begin{Bmatrix}x_i\\y_i\\z_i\end{Bmatrix}_{\mathrm{B}}+\sum N_i(\xi,\eta)\frac{(1-\zeta)}{2}\begin{Bmatrix}x_i\\y_i\\z_i\end{Bmatrix}_{\mathrm{T}} \tag{7-158}$$

令 $\Delta x_i,\Delta y_i,\Delta z_i$ 为节点 i 在厚度方向的坐标差值

$$\begin{Bmatrix}\Delta x_i\\\Delta y_i\\\Delta z_i\end{Bmatrix}=\begin{Bmatrix}x_i\\y_i\\z_i\end{Bmatrix}_{\mathrm{B}}-\begin{Bmatrix}x_i\\y_i\\z_i\end{Bmatrix}_{\mathrm{T}} \tag{7-159}$$

代入式(7-158)，得到单元内任一点的整体坐标如下

$$\begin{Bmatrix}x\\y\\z\end{Bmatrix}=\sum N_i\begin{Bmatrix}x_i\\y_i\\z_i\end{Bmatrix}_{\mathrm{M}}+\sum N_i\frac{\zeta}{2}\begin{Bmatrix}\Delta x_i\\\Delta y_i\\\Delta z_i\end{Bmatrix} \tag{7-160}$$

在式(7-158)～式(7-160)中，节点坐标向量中的下标 T 表示下表面，表明此时的节点坐标是与该节点相对应单元下表面处的坐标，类似地，B 表示上表面，M 表示中面，这样定义出近似的垂直于中面的局部坐标.

7.7.2 位移函数

在每一节点i处有3个线位移和两个角位移，为了定义角位移，如图7.5所示，作3个正交向量$\boldsymbol{H}_{1i},\boldsymbol{H}_{2i},\boldsymbol{H}_{3i}$。

首先，作垂直于中面的正交向量$\boldsymbol{H}_{3i}$

$$\boldsymbol{H}_{3i}=\{\Delta x_i \quad \Delta y_i \quad \Delta z_i\}^{\mathrm{T}} \tag{7-161}$$

$\boldsymbol{H}_{3i}$的方向余弦为

$$l_{3i}=\frac{\Delta x_i}{t_i},\ m_{3i}=\frac{\Delta y_i}{t_i},\ n_{3i}=\frac{\Delta z_i}{t_i}, \tag{7-162}$$

式中，$t_i=\left(\Delta x_i^2+\Delta y_i^2+\Delta z_i^2\right)^{1/2}$为节点$i$的壳体厚度。在节点$i$处再作两个向量$\boldsymbol{H}_{1i}$和$\boldsymbol{H}_{2i}$，它们均切于中面，并正交于$\boldsymbol{H}_{3i}$。为了使$\boldsymbol{H}_{1i},\boldsymbol{H}_{2i},\boldsymbol{H}_{3i}$与方向$x,y,z$大体一致，采用如下做法，令

$$\boldsymbol{H}_{2i}=\boldsymbol{H}_{3i}\times\boldsymbol{i},\ \boldsymbol{H}_{1i}=\boldsymbol{H}_{2i}\times\boldsymbol{H}_{3i} \tag{7-163}$$

即$\boldsymbol{H}_{2i}$正交于$\boldsymbol{H}_{3i}$和x轴，$\boldsymbol{H}_{1i}$正交于$\boldsymbol{H}_{2i}$和$\boldsymbol{H}_{3i}$，根据向量运算法则不难由式(7-163)求出$\boldsymbol{H}_{1i}$的方向余弦l_{1i},m_{1i},n_{1i}和$\boldsymbol{H}_{2i}$的方向余弦l_{2i},m_{2i},n_{2i}。

如图7.5所示，节点i在x,y,z方向的线位移分量分别为u_i,v_i,w_i。角位移如图7.6所示，ϕ_i为壳体中面法线向量$\boldsymbol{H}_{3i}$绕向量$\boldsymbol{H}_{2i}$的转角，ψ_i为向量$\boldsymbol{H}_{3i}$绕向量$\boldsymbol{H}_{1i}$的转角。

由于转角ϕ_i在$\boldsymbol{H}_{1i}$方向产生的线位移为$\frac{t_i\zeta\phi_i}{2}$，它在x,y,z方向的位移分量为$\frac{t_i\zeta\phi_i}{2}l_{1i},\frac{t_i\zeta\phi_i}{2}m_{1i},\frac{t_i\zeta\phi_i}{2}n_{1i}$，由于转角$\psi_i$在$\boldsymbol{H}_{2i}$方向产生的线位移为$\frac{t_i\zeta\psi_i}{2}$，它在$x,y,z$方向的位移分量为$\frac{t_i\zeta\psi_i}{2}l_{1i},\frac{t_i\zeta\psi_i}{2}m_{1i},\frac{t_i\zeta\psi_i}{2}n_{1i}$。

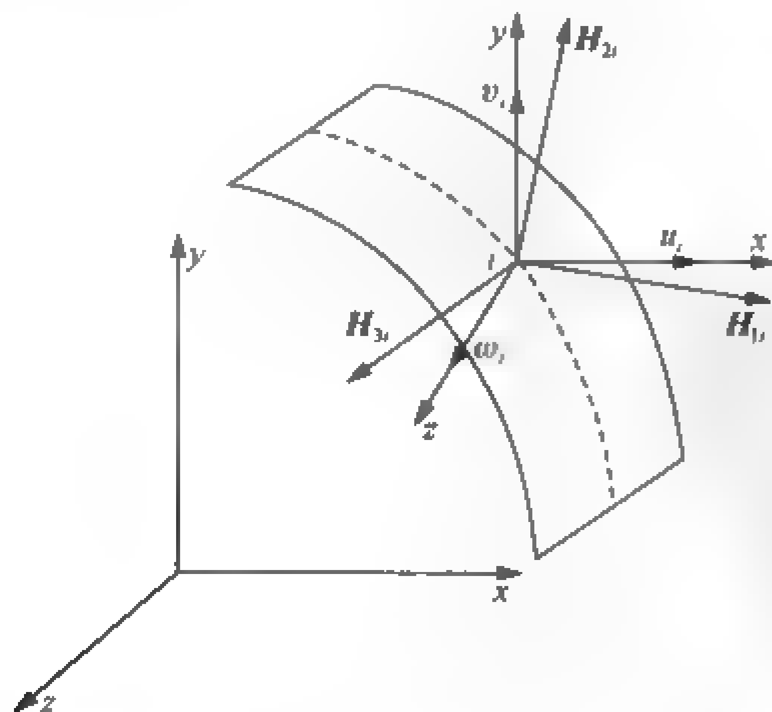


图7.5　线位移

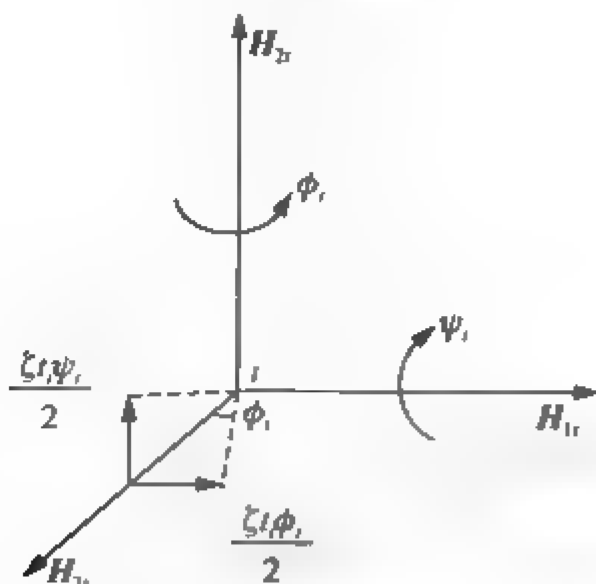


图7-6 角位移

利用形函数 $N_i(\xi,\eta)$，单元内任意一点的位移可以用中面节点位移 $\boldsymbol{\delta}^{\mathrm{T}}=[u_i,v_i,w_i,\phi_i,\psi_i]$ 表示如下

$$\begin{Bmatrix} u \\ v \\ w \end{Bmatrix}=\sum N_i\begin{Bmatrix} u_i \\ v_i \\ w_i \end{Bmatrix}+\sum\frac{\zeta N_i t_i}{2}\begin{bmatrix} l_{1i} & l_{2i} \\ m_{1i} & m_{2i} \\ n_{1i} & n_{2i} \end{bmatrix}\begin{Bmatrix} \phi_i \\ \psi_i \end{Bmatrix} \tag{7-164}$$

比较式(7-160)和式(7-164)可见，定义单元几何尺寸的式(7-160)比定义单元位移的式(7-164)具有较多的自由度，因此这种单元属于超参数单元，不会自动满足常应变条件. 但从后面对应变的定义中可以看出，关于刚体位移和常应变的条件是可以满足的.

7.7.3 整体坐标中的应变

整体坐标 (x,y,z) 中的应变取决于下列矩阵

$$\begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial v}{\partial x} & \frac{\partial w}{\partial x} \\ \frac{\partial u}{\partial y} & \frac{\partial v}{\partial y} & \frac{\partial w}{\partial y} \\ \frac{\partial u}{\partial z} & \frac{\partial v}{\partial z} & \frac{\partial w}{\partial z} \end{bmatrix} \tag{7-165}$$

将位移函数式(7-164)代入分别对 x,y,z 求微商，可得

$$\begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial v}{\partial x} & \frac{\partial w}{\partial x} \\ \frac{\partial u}{\partial y} & \frac{\partial v}{\partial y} & \frac{\partial w}{\partial y} \\ \frac{\partial u}{\partial z} & \frac{\partial v}{\partial z} & \frac{\partial w}{\partial z} \end{bmatrix}=\boldsymbol{J}^{-1}\begin{bmatrix} \frac{\partial u}{\partial \xi} & \frac{\partial v}{\partial \xi} & \frac{\partial w}{\partial \xi} \\ \frac{\partial u}{\partial \eta} & \frac{\partial v}{\partial \eta} & \frac{\partial w}{\partial \eta} \\ \frac{\partial u}{\partial \zeta} & \frac{\partial v}{\partial \zeta} & \frac{\partial w}{\partial \zeta} \end{bmatrix} \tag{7-166}$$

其中 $\boldsymbol{J}$ 为雅可比矩阵

$$J=\begin{bmatrix}\frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi}\\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta}\\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta}\end{bmatrix} \tag{7-167}$$

7.7.4 局部坐标中的应变与应力

在厚壳的分析中，由于壳体中面的法线方向与整体坐标系的z轴并不一致，而且法线的方向随着点的位置不同而变化，既然假定垂直于壳体中面的正应力等于零，就必须求出局部坐标系中的应力与应变. 现假定局部坐标为(x',y',z')，其中z'轴正交于壳体中面.

前面已经求出了节点i的局部正交向量$\boldsymbol{H}_{1i},\boldsymbol{H}_{2i},\boldsymbol{H}_{3i}$. 现在建立单元内任意一点的局部坐标$(x',y',z')$.

如图 7.7 所示，在(ξ_0,η_0,ζ_0)点作曲面$\zeta=\zeta_0$，在此曲面上有两条空间曲线，一条是$\eta=\eta_0$，另一条是$\xi=\xi_0$. 再作曲面$\zeta=\zeta_0$的两个切向量$\mathrm{d}\boldsymbol{\xi}$和$\mathrm{d}\boldsymbol{\eta}$，其中$\mathrm{d}\boldsymbol{\xi}$切于曲线$\eta=\eta_0$，$\mathrm{d}\boldsymbol{\eta}$切于曲线$\xi=\xi_0$，由此可知，$\mathrm{d}\boldsymbol{\xi}=\left(\boldsymbol{i}\frac{\partial x}{\partial \xi}+\boldsymbol{j}\frac{\partial y}{\partial \xi}+\boldsymbol{k}\frac{\partial z}{\partial \xi}\right)\mathrm{d}\xi$，$\mathrm{d}\boldsymbol{\eta}=\left(\boldsymbol{i}\frac{\partial x}{\partial \eta}+\boldsymbol{j}\frac{\partial y}{\partial \eta}+\boldsymbol{k}\frac{\partial z}{\partial \eta}\right)\mathrm{d}\eta$. 式中$\boldsymbol{i}$，$\boldsymbol{j}$，$\boldsymbol{k}$为$x$，$y$，$z$方向的单位向量.

作z'垂直于$\mathrm{d}\boldsymbol{\xi}$和$\mathrm{d}\boldsymbol{\eta}$，即正交于$\zeta=\zeta_0$曲面

$$z'=\mathrm{d}\boldsymbol{\xi}\times\mathrm{d}\boldsymbol{\eta}=\begin{vmatrix}\boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k}\\ \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi}\\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\nabla \eta}\end{vmatrix} \tag{7-168}$$

由此可以求出z'的方向余弦l_3,m_3,n_3. 为使局部坐标(x',y',z')与整体坐标(x,y,z)大体一致，以便于结果的整理和边界条件的处理，对于其他两个坐标x'和y'用如下方法选取. 令

$$y'=z'\times x=\begin{vmatrix}\boldsymbol{i} & \boldsymbol{i} & \boldsymbol{k}\\ l_3 & m_3 & n_3\\ 1 & 0 & 0\end{vmatrix} \tag{7-169}$$

由此求出y'的方向余弦l_2,m_2,n_2. 如果z'轴和x轴平行，可改用$y'=z'\times y$. 最后，作x'正交于y'和z'，即$x'=y'\times z'$，由此可以求出x'的方向余弦l_1,m_1,n_1.

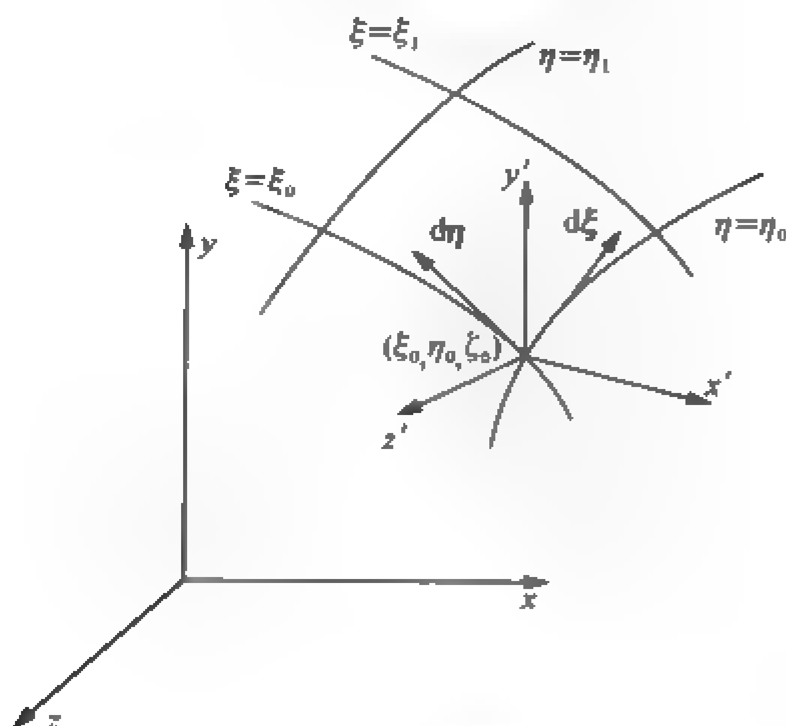


图 7.7 局部坐标系

现在得到局部坐标(x',y',z')的方向余弦矩阵如下

$$\boldsymbol{\theta}=\begin{bmatrix} l_1 & l_2 & l_3 \\ m_1 & m_2 & m_3 \\ n_1 & n_2 & n_3 \end{bmatrix} \tag{7-170}$$

利用矩阵$\boldsymbol{\theta}$可将整体坐标系中的矩阵式(7-165)转换到局部坐标系中去

$$\begin{bmatrix} \frac{\partial u'}{\partial x'} & \frac{\partial v'}{\partial x'} & \frac{\partial w'}{\partial x'} \\ \frac{\partial u'}{\partial y'} & \frac{\partial v'}{\partial y'} & \frac{\partial w'}{\partial y'} \\ \frac{\partial u'}{\partial z'} & \frac{\partial v'}{\partial z'} & \frac{\partial w'}{\partial z'} \end{bmatrix}=\boldsymbol{\theta}^{\mathrm{T}}\begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial v}{\partial x} & \frac{\partial w}{\partial x} \\ \frac{\partial u}{\partial y} & \frac{\partial v}{\partial y} & \frac{\partial w}{\partial y} \\ \frac{\partial u}{\partial z} & \frac{\partial v}{\partial z} & \frac{\partial w}{\partial z} \end{bmatrix}\boldsymbol{\theta} \tag{7-171}$$

由此得到局部坐标系中的应变如下

$$\boldsymbol{\varepsilon}'=\begin{Bmatrix} \varepsilon'_x \\ \varepsilon'_y \\ \gamma'_{xy} \\ \gamma'_{yz} \\ \gamma'_{zx} \end{Bmatrix}=\begin{Bmatrix} \frac{\partial u'}{\partial x'} \\ \frac{\partial v'}{\partial y'} \\ \frac{\partial u'}{\partial y'}+\frac{\partial v'}{\partial x'} \\ \frac{\partial w'}{\partial y'}+\frac{\partial v'}{\partial z'} \\ \frac{\partial w'}{\partial x'}+\frac{\partial u'}{\partial z'} \end{Bmatrix}=\boldsymbol{B}\boldsymbol{\delta} \tag{7-172}$$

将矩阵$\boldsymbol{B}$分块，得到

$$\boldsymbol{\varepsilon}'=[B_1,\cdots,B_i,\cdots B_8]\begin{Bmatrix}\delta_1\\ \vdots\\ \delta_i\\ \vdots\\ \delta_8\end{Bmatrix}=\sum \boldsymbol{B}_i\boldsymbol{\delta}_i \tag{7-173}$$

式中：$\boldsymbol{\delta}_i^{\mathrm{T}}=\{u_i,v_i,w_i,\phi_i,\psi_i\}$；$\boldsymbol{B}_i$ 为如下5×5矩阵

$$\boldsymbol{B}_i=\begin{bmatrix} l_1\alpha_1 & m_1\alpha_1 & n_1\alpha_1 & \beta_1\gamma_1 & \beta_1\lambda_1\\ l_2\alpha_2 & m_2\alpha_2 & n_2\alpha_2 & \beta_2\gamma_2 & \beta_2\lambda_2\\ l_1\alpha_2+l_2\alpha_1 & m_1\alpha_2+m_2\alpha_1 & n_1\alpha_2+n_2\alpha_1 & \beta_1\gamma_2+\beta_2\gamma_1 & \beta_1\lambda_2+\beta_2\lambda_1\\ l_2\alpha_3+l_3\alpha_2 & m_2\alpha_3+m_3\alpha_2 & n_2\alpha_3+n_3\alpha_2 & \beta_2\gamma_3+\beta_3\gamma_2 & \beta_2\lambda_3+\beta_3\lambda_2\\ l_3\alpha_1+l_1\alpha_3 & m_3\alpha_1+m_1\alpha_3 & n_3\alpha_1+n_1\alpha_3 & \beta_3\gamma_1+\beta_1\gamma_3 & \beta_3\lambda_1+\beta_1\lambda_3 \end{bmatrix} \tag{7-174}$$

其中

$$\left.\begin{aligned} &\alpha_s=l_s\frac{\partial N_i}{\partial x}+m_s\frac{\partial N_i}{\partial y}+n_s\frac{\partial N_i}{\partial z}\\ &\beta_s=\left(l_s\frac{\partial M_i}{\partial x}+m_s\frac{\partial M_i}{\partial y}+n_s\frac{\partial M_i}{\partial z}\right)\frac{t_i}{2}\\ &\gamma_s=l_sl_{1i}+m_sm_{1i}+n_sn_{1i}\\ &\lambda_s=l_sl_{2i}+m_sm_{2i}+n_sn_{2i} \end{aligned}\right\} \tag{7-175}$$

其中 $M_i=N_i\zeta$.

在应力－应变关系式中令 $\sigma_z=0$，可得各向同性弹性固体的5×5弹性矩阵如下

$$\boldsymbol{D}=\frac{E}{1-\mu^2}\begin{bmatrix} 1 & \mu & 0 & 0 & 0\\ \mu & 1 & 0 & 0 & 0\\ 0 & 0 & \frac{1-\mu}{2} & 0 & 0\\ 0 & 0 & 0 & \frac{1-\mu}{2k} & 0\\ 0 & 0 & 0 & 0 & \frac{1-\mu}{2k} \end{bmatrix} \tag{7-176}$$

式中：E 和 μ 为弹性模量和泊松比；最后两个剪应力项中包含的系数 k 可取为 1.20. 引入系数 k 是为了考虑剪应力分布不均匀的影响. 根据前面所述位移函数，剪应力沿厚度方向接近均匀分布，实际上是抛物线分布，k 的数值就是两种应变能的比值.

由广义胡克定理，局部坐标系中的单元应力为

$$\boldsymbol{\sigma}'=\begin{Bmatrix}\sigma_{x'}\\ \sigma_{y'}\\ \tau_{x'y'}\\ \tau_{y'z'}\\ \tau_{z'x'}\end{Bmatrix}=\boldsymbol{D}(\boldsymbol{\varepsilon}'-\boldsymbol{\varepsilon}_0')+\boldsymbol{\sigma}_0' \tag{7-177}$$

式中：$\boldsymbol{\varepsilon}_0'$为初应变；$\boldsymbol{\sigma}_0'$为初应力；$\boldsymbol{D}$为弹性矩阵. 在刚度系数计算中，直接利用局部坐标系中的应力$\boldsymbol{\sigma}'$，在计算输入结果时，除$\boldsymbol{\sigma}'$外还应输出整体坐标系中的应力.

$$\begin{bmatrix} \sigma_x & \tau_{xy} & \tau_{xz} \\ \tau_{xy} & \sigma_y & \tau_{yz} \\ \tau_{xz} & \tau_{yz} & \sigma_z \end{bmatrix} = \boldsymbol{\theta} \begin{bmatrix} \sigma_{x'} & \tau_{x'y'} & \tau_{x'z'} \\ \tau_{x'y'} & \sigma_{y'} & \tau_{y'z'} \\ \tau_{x'z'} & \tau_{y'z'} & 0 \end{bmatrix} \boldsymbol{\theta}^{T} \tag{7-178}$$

7.7.5 单元刚度矩阵与节点载荷

单元刚度矩阵直接在局部坐标系中使用下式计算

$$\boldsymbol{k}_{rs} = \iiint \boldsymbol{B}_r^{\mathrm{T}} \boldsymbol{D} \boldsymbol{B}_s |\boldsymbol{J}| \mathrm{d}\xi \mathrm{d}\eta \mathrm{d}\zeta \tag{7-179}$$

可以通过数值积分来求出刚度矩阵. 需要指出的是，正如轴对称壳体情况，超参数单元在引入一定的几何假设后和位移及转角各自独立插值的壳单元是等价的. 进一步的研究工作指出，对于作为三维蜕化实体元的超参单元，式(7-172)中所表达的应变分量$\varepsilon_{x'}, \varepsilon_{y'}$和$\gamma_{x'y'}$实际上包含着沿厚度(即局部坐标系$z'$)均匀分布和线性分布两部分. 后者是壳体的弯曲应变，它相当于 Mindlin 板中的曲率变化κ引起的应变；前者是壳体的薄膜应变，是壳体中面内的变形(在板弯曲问题中，它被忽略)所引起的. 通过量纲分析可以认为，薄膜应变能项和剪切应变能项相同，在泛函中也具有罚函数的性质. 因此与它相关的刚度矩阵也应是奇异的. 否则，在壳体越来越薄时，它也会造成“锁死”现象. 因为这种锁死是由过分的虚假薄膜应变能引起的，所以称之为薄膜“锁死”.

为了保证非奇异性，且避免剪切“锁死”和薄膜“锁死”，从原则上说，在 Mindlin 板单元中讨论的各种方法都可以用于这种超参数单元. 例如对$\varepsilon_{x'}, \varepsilon_{y'}$和$\gamma_{x'y'}$的弯曲应变部分和薄膜应变部分以及横剪应变$\gamma_{x'z'}$和$\gamma_{y'z'}$采用不同取样点的插值表示. 但是对于一般形状的超参数单元，整个推导过程和表达式相当复杂. 因此从实用角度考虑，一般采用的方法是在对式(7-179)进行数值积分时使用缩减积分的方案.

数值积分是在积分区间内选取一定的积分点，累加这些积分点上的函数值与相应系数的乘积得到积分的近似值. 根据选择积分点的位置和系数计算方法的不同，数值积分也包括多种方法，例如牛顿－科斯特积分和高斯积分等. 高斯积分法能在同样的积分点数目下实现更高的精度，是一般情况下采用的数值积分方法. 在使用数值积分时，首先要选择积分点的数目，较多的点数可以获得更高的精度. 但是分析表明，在对式(7-179)进行数值积分时，如果ξ,η方向采用3×3点的积分，在对厚板和厚壳进行计算时，能得到较好的结果；而用于薄板和薄壳时，结果反而不好. 其原因是在厚度变小时，发生了“锁死”现象. 而改用2×2点的积分方案，则可以避免“锁死”现象，精度可以得到明显的改进，且节省了计算时间. 所以对本节中的八节点单元，可以在ξ,η和ζ方向都采用这样的 2 点缩减积分方案.

节点力与节点位移之间存在如下关系

$$F_r=\begin{Bmatrix}U_r\\V_r\\W_r\\M_r^{\varphi}\\N_r^{\psi}\end{Bmatrix}=\sum_{s=1}^{8}\begin{bmatrix}k_{rs}^{11} & k_{rs}^{12} & \cdots & k_{rs}^{15}\\k_{rs}^{21} & k_{rs}^{22} & \cdots & k_{rs}^{25}\\\vdots & \vdots & \vdots & \vdots\\k_{rs}^{51} & k_{rs}^{52} & \cdots & k_{rs}^{55}\end{bmatrix}\begin{Bmatrix}u_s\\v_s\\w_s\\\varphi_s\\\psi_s\end{Bmatrix} \tag{7-180}$$

设壳体中面承受了分布载荷 p，在节点 i 分布载荷在 x, y, z 方向的分量分别为 p_i^x, p_i^y, p_i^z 中面上任一点的分布载荷可表示为

$$p_x=\sum N_i p_i^x,\quad p_y=\sum N_i p_i^y,\quad p_z=\sum N_i p_i^z \tag{7-181}$$

在曲面 $\zeta=0$ 上取向量 $\mathrm{d}\boldsymbol{\xi}$ 和 $\mathrm{d}\boldsymbol{\eta}$ 所构成的微小面积 $\mathrm{d}\Omega$，由向量运算法则可知：$\mathrm{d}\Omega=|\mathrm{d}\boldsymbol{\xi}\times\mathrm{d}\boldsymbol{\eta}|=|A|\mathrm{d}\xi\mathrm{d}\eta$，其中

$$|A|=\left\{\left(\frac{\partial x}{\partial\xi}-\frac{\partial y}{\partial\eta}-\frac{\partial x}{\partial\eta}-\frac{\partial y}{\partial\xi}\right)^2+\left(\frac{\partial y}{\partial\xi}-\frac{\partial z}{\partial\eta}-\frac{\partial y}{\partial\eta}-\frac{\partial z}{\partial\xi}\right)^2\right.$$
$$\left.+\left(\frac{\partial z}{\partial\eta}-\frac{\partial x}{\partial\xi}-\frac{\partial z}{\partial\xi}-\frac{\partial x}{\partial\eta}\right)^2\right\}^{\frac{1}{2}} \tag{7-182}$$

由虚功原理，得到分布载荷 p 产生的节点载荷如下

$$\left.\begin{aligned}X_r^p&=\iint N_r\left(\sum N_i p_i^x\right)\cdot|A|\mathrm{d}\xi\mathrm{d}\eta\\Y_r^p&=\iint N_r\left(\sum N_i p_i^x y\right)\cdot|A|\mathrm{d}\xi\mathrm{d}\eta\\Z_r^p&=\iint N_r\left(\sum N_i p_i^z\right)\cdot|A|\mathrm{d}\xi\mathrm{d}\eta\\M_r^{\varphi p}&=0,\qquad M_r^{\psi p}=0\end{aligned}\right\} \tag{7-183}$$

7.7.6 单元质量矩阵

对这种壳体单元可以采用如下集中质量矩阵：整个单元质量为 W，每个节点集中单元 1/8 的质量，并略去转动项，得到 40×40 阶的矩阵 $\boldsymbol{M}$

$$\boldsymbol{M}=\begin{bmatrix}\boldsymbol{m} & 0 & \cdots & 0\\0 & \boldsymbol{m} & \cdots & 0\\\vdots & \vdots & \vdots & \vdots\\0 & 0 & \cdots & \boldsymbol{m}\end{bmatrix} \tag{7-184}$$

在 $\boldsymbol{M}$ 的对角线上的 8 个子矩阵 $\boldsymbol{m}$ 是 5×5 的对角矩阵.

$$\boldsymbol{m}=\frac{W}{8}\begin{bmatrix}1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 0 & \\ & & & & 0\end{bmatrix} \tag{7-185}$$

7.8 复合材料单元

因为复合材料结构和一般结构相比，主要区别是材料性质. 所以上面介绍的各种单元，包括本书其他章节介绍的各种单元，通过修改 $\boldsymbol{K}^e=\int_{Ve}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{B}\mathrm{d}V$ 中的弹性矩阵 $\boldsymbol{D}$，一般都可以用来计算复合材料. 对单元其他部分一般不需要做特别改动. 因此，在这里只介绍应用 7.7 节中介绍的壳体单元研究多层复合材料结构时，计算叠层弹性矩阵 $\boldsymbol{D}''$ 的过程. 对于更简单的板单元，可以使用类似的计算过程，这里不再叙述.

在复合材料层合壳体中，由于复合材料是各向异性材料，所以其第 i 层主轴弹性矩阵 $\boldsymbol{Q}_i$ 不同于式(7-176)中各项同性材料的弹性矩阵. $\boldsymbol{Q}_i$ 可由主轴的弹性模量 $E_1^{(i)}$、$E_2^{(i)}$、剪切模量 $G^{(i)}$ 和材料的泊松比 $\mu_{12}^{(i)}$、$\mu_{21}^{(i)}$ 表示为

$$\boldsymbol{Q}_i=\begin{bmatrix}\dfrac{E_1^{(i)}}{1-\mu_{12}^{(i)}\mu_{21}^{(i)}} & \dfrac{\mu_{12}^{(i)}E_1^{(i)}}{1-\mu_{12}^{(i)}\mu_{21}^{(i)}} & 0 & 0 & 0\\ \dfrac{\mu_{21}^{(i)}E_2^{(i)}}{1-\mu_{12}^{(i)}\mu_{21}^{(i)}} & \dfrac{E_2^{(i)}}{1-\mu_{12}^{(i)}\mu_{21}^{(i)}} & 0 & 0 & 0\\ 0 & 0 & G^{(i)} & 0 & 0\\ 0 & 0 & 0 & \dfrac{G^{(i)}}{k} & 0\\ 0 & 0 & 0 & 0 & \dfrac{G^{(i)}}{k}\end{bmatrix} \tag{7-186}$$

同样引入系数 $k=1.2$ 是为了考虑剪应力分布不均匀的影响.

由于每一层材料的主轴方向一般与坐标轴方向不重合，第 i 层主轴应变 $\bar{\boldsymbol{\varepsilon}}_i$、应力 $\bar{\boldsymbol{\sigma}}_i$、与坐标轴方向的应力 $\boldsymbol{\sigma}_i$、应变 $\boldsymbol{\varepsilon}_i$ 有以下变换关系

$$\left.\begin{aligned}\boldsymbol{\varepsilon}_i&=\boldsymbol{T}_i^{-\mathrm{T}}\bar{\boldsymbol{\varepsilon}}_i\\ \boldsymbol{\sigma}_i&=\boldsymbol{T}_i\bar{\boldsymbol{\sigma}}_i\end{aligned}\right\} \tag{7-187}$$

式中，偏轴应力变化矩阵 $\boldsymbol{T}_i$ 为

$$\boldsymbol{T}_i=\begin{bmatrix}\cos^2\theta_i & \sin^2\theta_i & 2\sin\theta_i\cos\theta_i & 0 & 0\\ \sin^2\theta_i & \cos^2\theta_i & -2\sin\theta_i\cos\theta_i & 0 & 0\\ -\sin\theta_i\cos\theta_i & \sin\theta_i\cos\theta_i & \cos^2\theta_i-\sin^2\theta_i & 0 & 0\\ 0 & 0 & 0 & \cos\theta_i & -\sin\theta_i\\ 0 & 0 & 0 & \sin\theta_i & \cos\theta_i\end{bmatrix} \tag{7-188}$$

θ_i 为第 i 层的主轴偏角.

复合材料第 i 层主轴应变 $\bar{\boldsymbol{\varepsilon}}_i$ 和应力 $\bar{\boldsymbol{\sigma}}_i$ 的关系为

$$\bar{\boldsymbol{\sigma}}_i=\boldsymbol{Q}_i\bar{\boldsymbol{\varepsilon}}_i \tag{7-189}$$

综合式(7-187)和式(7-189)可得坐标轴方向的应力$\boldsymbol{\varepsilon}_i$和应变$\boldsymbol{\sigma}_i$的关系为

$$\boldsymbol{\sigma}_i=\bar{\boldsymbol{Q}}_i\boldsymbol{\varepsilon}_i \tag{7-190}$$

其中

$$\bar{\boldsymbol{Q}}_i=\boldsymbol{T}_i\boldsymbol{Q}_i\boldsymbol{T}_i^{\mathrm{T}} \tag{7-191}$$

需要注意的是，对于复合材料层合壳体，除圆柱壳体外，其他壳体的$\boldsymbol{Q}_i$将不是常数而是x',y'的函数. 这是因为在制造复合材料层合壳体时(除圆柱壳体外)，不论采用什么制造方法，至少存在如下两种情形之一：沿壳面单位宽度的纤维密度不一样，使得$\boldsymbol{Q}_i$沿壳面有变化；纤维方向与壳体上曲率方向的夹角θ不是常数. 这样，$\boldsymbol{Q}_i$不但沿厚度方向是不均匀的，而且沿壳面方向也是不均匀的，且其变化关系与制造方法有关，难以确定. 所以一般形状的复合材料层合壳体的分析是十分复杂的. 假设$\boldsymbol{Q}_i$与x',y'无关，对一般壳体情况是不精确的，对锥角不大的圆锥壳和很扁的扁壳则是很好的近似，只有对圆柱壳是精确的.

在得到各层的弹性矩阵后，可以计算出层合壳体情况下与式(7-179)中$\boldsymbol{D}$相对应的叠层弹性矩阵$\boldsymbol{D}^{m}$. 对于一般壳体单元，$\boldsymbol{D}^{m}$由拉伸刚度$\bar{\boldsymbol{A}}$矩阵(extensional stiffness matrices)、耦合刚度矩阵$\bar{\boldsymbol{B}}$(extensional-bending coupling stiffness matrices)、弯曲刚度矩阵$\bar{\boldsymbol{D}}$(bending stiffness matrices)和横向剪力刚度矩阵$\bar{\boldsymbol{A}}_s$(transverse shear stiffness matrices)构成. 对于一般工程中使用的对称叠层的层合材料，其耦合刚度$\bar{\boldsymbol{B}}$为零. 而且如前所述，在这种壳体单元中的应变分量$\varepsilon_{x'}$，$\varepsilon_{y'}$和$\gamma_{x'y'}$实际上已经包含了壳体的薄膜应变和壳体的弯曲应变两部分. 所以不必再考虑耦合刚度$\bar{\boldsymbol{B}}$和弯曲刚度$\bar{\boldsymbol{D}}$.

$$\bar{\boldsymbol{A}}=\sum_{k=1}^{n}\begin{bmatrix}\bar{Q}_{11} & \bar{Q}_{12} & \bar{Q}_{13}\\ \bar{Q}_{21} & \bar{Q}_{22} & \bar{Q}_{23}\\ \bar{Q}_{31} & \bar{Q}_{31} & \bar{Q}_{33}\end{bmatrix}_k(z_k-z_{k-1})=\sum_{k=1}^{n}\begin{bmatrix}\bar{Q}_{11} & \bar{Q}_{12} & \bar{Q}_{13}\\ \bar{Q}_{21} & \bar{Q}_{22} & \bar{Q}_{23}\\ \bar{Q}_{31} & \bar{Q}_{31} & \bar{Q}_{33}\end{bmatrix}_k t_k$$

$$\begin{aligned}\bar{\boldsymbol{A}}_s&=\frac{5}{4}\int_{-\frac{h}{2}}^{\frac{h}{2}}\begin{bmatrix}\bar{Q}_{44} & \bar{Q}_{45}\\ \bar{Q}_{54} & \bar{Q}_{55}\end{bmatrix}_k\left(1-\frac{4z^2}{h^2}\right)\mathrm{d}z\\ &=\sum_{k=1}^{n}\begin{bmatrix}\bar{Q}_{44} & \bar{Q}_{45}\\ \bar{Q}_{54} & \bar{Q}_{55}\end{bmatrix}_k\left(\frac{5}{4}t_k-\frac{5}{3t_k^2}\left(z_{x+1}^3-z_x^3\right)\right)\end{aligned} \tag{7-192}$$

式中：n是材料的层数；z_k是第k层外侧一面距离壳体中面的距离；t_k是第k层的厚度. $\left[\bar{Q}_{ij}\right]_k$是指第$k$层的弹性矩阵$\bar{\boldsymbol{Q}}_k$中的第$i$行第$j$列的元素. h是壳体厚度.

则叠层弹性矩阵$\boldsymbol{D}^{m}$为

$$\boldsymbol{D}^{m}=\begin{bmatrix}\bar{\boldsymbol{A}} & \boldsymbol{0}\\ \boldsymbol{0} & \bar{\boldsymbol{A}}_s\end{bmatrix} \tag{7-193}$$

此时单元刚度矩阵为

$$\boldsymbol{k}_{rs}=\iiint\boldsymbol{B}_r^{\mathrm{T}}\boldsymbol{D}^{m}\ \ \boldsymbol{B}_s|\boldsymbol{J}|\mathrm{d}\xi\mathrm{d}\eta\mathrm{d}\zeta \tag{7-194}$$

7.9　应用问题与 MATLAB 程序

7.9.1　求一边固支方板的频率

求如图 7.8 所示的一边固支方板的前 15 阶固有频率. 该板的长度和宽度都为 2m，厚度为 0.05m，材料弹性模量 $E=2.1\text{GPa}$，泊松比 $\mu=0.3$，密度 $\rho=7\,300\text{Kg}/\text{m}^3$.

使用四节点的经典板单元来求解. 将板划分为 100 个单元. 在此例中，因为所有单元的大小和形状都一样，各单元的单元刚度矩阵和质量矩阵相同，只需计算出一个单元的单元刚度矩阵和质量矩阵即可，不必挨个计算每个单元. 所以在程序中没有一般有限元程序中所必须有的节点坐标数组.

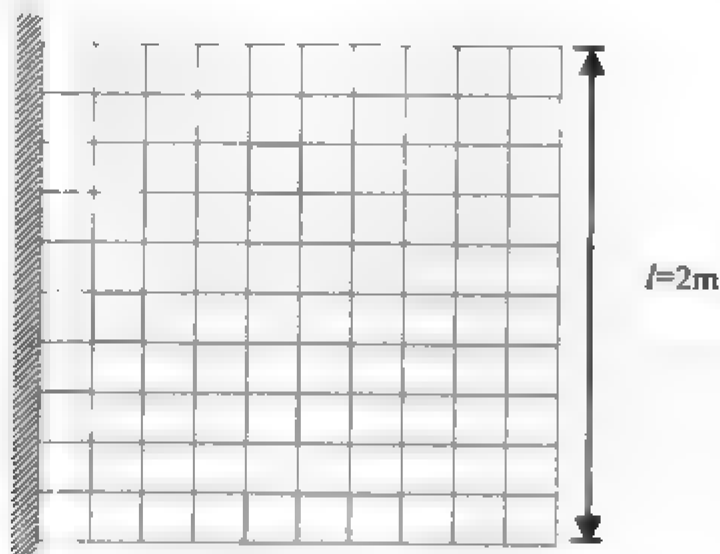


图 7.8　一边固支方板

程序如下：

```
E=2.1e11;        %elastic molulus
poisson =0.3;        % poisson ratio
density=7.3e3;       %density
t=0.05;             %plate thickness
lx=2;               %length in x direction
ly=2;               %length in y direction
jdx=11;             %number of nodes in x direction
jdy=11;             %number of nodes in y direction

k(1:330,1:330)=0;     %system stiffness matrix
m(1:330,1:330)=0;    %system mass matrix

%prepare the arrays which are needed to describe this problem
en(1:100,1:4)=0;      %element node
for ni=1:jdx-1
   for nj=1:jdy-1
      en(ni+(nj-1)*(jdx-1),1)=ni+(nj-1)*jdx;
      en(ni+(nj-1)*(jdx-1),2)=ni+1+(nj-1)*jdx;
```

```
        en(ni+(nj-1)*(jdx-1),4)=ni+nj*jdx;
        en(ni+(nj-1)*(jdx-1),3)=ni+1+nj*jdx;
    end
end
disp(1:jdx*jdy,1:3)=1;    % node displacement
constraints=1:jdx:jdx*jdy;  % constraints
disp(constraints,:)=0;
dof=0;                  %degree of freedom
for ni=1:jdx*jdy
    for nj=1:3
        if disp(ni,nj)~=0
            dof=dof+1;
            disp(ni,nj)=dof;
        end
    end
end

el=lx/(jdx-1);        %element length
eh=ly/(jdy-1);       %element height
[ek,dm]=km(el/2,eh/2,mu,poisson,E,density);
%km: function used to compute element stifness and mass matrix
%in this case, all elements have the same element stifness and mass
matrix.

%built system stifness and mass matrix.
index(1:12)=0; % vector sontaining system dofs of nodes in each element.
for loopi=1:(jdx-1)*(jdy-1)
    for zi=1:4
        index((zi-1)*3+1)=disp(en(loopi,zi),1);
        index((zi-1)*3+2)=disp(en(loopi,zi),2);
        index((zi-1)*3+3)=disp(en(loopi,zi),3);
    end
    for jx=1:12
        for jy=1:12
            if(index(jx)*index(jy)~=0)
                k(index(jx),index(jy))=k(index(jx),index(jy))+ek(jx,jy);
                m(index(jx),index(jy))=m(index(jx),index(jy))+em(jx,jy);
            end
        end
    end
end

%solve eigenvalue problem
[v,d] = eig(k,m);
tempd=diag(d);
[nd,sortindex]=sort(tempd);
v=v(:,sortindex);
```

```
mode_number=1:15;
frequency(mode_number)=sqrt(nd(mode_number))/(2*pi);

function [k,m]=km(a,b,poisson,t,E,density)
k=[E*t^3/(360-360*poisson^2)/a/b*(21-6*poisson+30*b^2/a^2+30*a^2/b^2),
E*t^3/(360-360*poisson^2)/a/b*(3*b+12*poisson*b+30*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(-3*a-12*poisson*a-30*b^2/a),
E*t^3/(360-360*poisson^2)/a/b*(-21+6*poisson-30*b^2/a^2+15*a^2/b^2),
E*t^3/(360-360*poisson^2)/a/b*(-3*b-12*poisson*b+15*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(-3*a+3*poisson*a-30*b^2/a),
E*t^3/(360-360*poisson^2)/a/b*(21-6*poisson-15*b^2/a^2-15*a^2/b^2),
E*t^3/(360-360*poisson^2)/a/b*(-3*b+3*poisson*b+15*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(3*a-3*poisson*a-15*b^2/a),
E*t^3/(360-360*poisson^2)/a/b*(-21+6*poisson+15*b^2/a^2-30*a^2/b^2),
E*t^3/(360-360*poisson^2)/a/b*(3*b-3*poisson*b+30*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(3*a+12*poisson*a-15*b^2/a);
E*t^3/(360-360*poisson^2)/a/b*(3*b+12*poisson*b+30*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(8*b^2-8*poisson*b^2+40*a^2),
-30*E*t^3/(360-360*poisson^2)*poisson,
E*t^3/(360-360*poisson^2)/a/b*(-3*b-12*poisson*b+15*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(-8*b^2+8*poisson*b^2+20*a^2),
0,
E*t^3/(360-360*poisson^2)/a/b*(3*b-3*poisson*b-15*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(2*b^2-2*poisson*b^2+10*a^2),
0,
E*t^3/(360-360*poisson^2)/a/b*(-3*b+3*poisson*b-30*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(-2*b^2+2*poisson*b^2+20*a^2),
0;
E*t^3/(360-360*poisson^2)/a/b*(-3*a-12*poisson*a-30*b^2/a),
-30*E*t^3/(360-360*poisson^2)*poisson,
E*t^3/(360-360*poisson^2)/a/b*(8*a^2-8*poisson*a^2+40*b^2),
E*t^3/(360-360*poisson^2)/a/b*(3*a-3*poisson*a+30*b^2/a),
0,
E*t^3/(360-360*poisson^2)/a/b*(-2*a^2+2*poisson*a^2+20*b^2),
E*t^3/(360-360*poisson^2)/a/b*(-3*a+3*poisson*a+15*b^2/a),
0,
E*t^3/(360-360*poisson^2)/a/b*(2*a^2-2*poisson*a^2+10*b^2),
E*t^3/(360-360*poisson^2)/a/b*(3*a+12*poisson*a-15*b^2/a),
0,
E*t^3/(360-360*poisson^2)/a/b*(-8*a^2+8*poisson*a^2+20*b^2);
E*t^3/(360-360*poisson^2)/a/b*(-21+6*poisson-30*b^2/a^2+15*a^2/b^2),
E*t^3/(360-360*poisson^2)/a/b*(-3*b-12*poisson*b+15*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(3*a-3*poisson*a+30*b^2/a),
E*t^3/(360-360*poisson^2)/a/b*(21-6*poisson+30*b^2/a^2+30*a^2/b^2),
E*t^3/(360-360*poisson^2)/a/b*(3*b+12*poisson*b+30*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(3*a+12*poisson*a+30*b^2/a),
E*t^3/(360-360*poisson^2)/a/b*(-21+6*poisson+15*b^2/a^2-30*a^2/b^2),
```

```
E*t^3/(360-360*poisson^2)/a/b*(3*b-3*poisson*b+30*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(-3*a-12*poisson*a+15*b^2/a),
E*t^3/(360-360*poisson^2)/a/b*(21-6*poisson-15*b^2/a^2-15*a^2/b^2),
E*t^3/(360-360*poisson^2)/a/b*(-3*b+3*poisson*b+15*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(-3*a+3*poisson*a+15*b^2/a);
E*t^3/(360-360*poisson^2)/a/b*(-3*b-12*poisson*b+15*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(-8*b^2+8*poisson*b^2+20*a^2),
0,
E*t^3/(360-360*poisson^2)/a/b*(3*b+12*poisson*b+30*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(8*b^2-8*poisson*b^2+40*a^2),
30*E*t^3/(360-360*poisson^2)*poisson,
E*t^3/(360-360*poisson^2)/a/b*(-3*b+3*poisson*b-30*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(-2*b^2+2*poisson*b^2+20*a^2),
0,
E*t^3/(360-360*poisson^2)/a/b*(3*b-3*poisson*b-15*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(2*b^2-2*poisson*b^2+10*a^2),
0;
E*t^3/(360-360*poisson^2)/a/b*(-3*a+3*poisson*a-30*b^2/a),
0,
E*t^3/(360-360*poisson^2)/a/b*(-2*a^2+2*poisson*a^2+20*b^2),
E*t^3/(360-360*poisson^2)/a/b*(3*a+12*poisson*a+30*b^2/a),
30*E*t^3/(360-360*poisson^2)*poisson,
E*t^3/(360-360*poisson^2)/a/b*(8*a^2-8*poisson*a^2+40*b^2),
E*t^3/(360-360*poisson^2)/a/b*(-3*a-12*poisson*a+15*b^2/a),
0,
E*t^3/(360-360*poisson^2)/a/b*(-8*a^2+8*poisson*a^2+20*b^2),
E*t^3/(360-360*poisson^2)/a/b*(3*a-3*poisson*a-15*b^2/a),
0,
E*t^3/(360-360*poisson^2)/a/b*(2*a^2-2*poisson*a^2+10*b^2);
E*t^3/(360-360*poisson^2)/a/b*(21-6*poisson-15*b^2/a^2-15*a^2/b^2),
E*t^3/(360-360*poisson^2)/a/b*(3*b-3*poisson*b-15*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(-3*a+3*poisson*a+15*b^2/a),
E*t^3/(360-360*poisson^2)/a/b*(-21+6*poisson+15*b^2/a^2-30*a^2/b^2),
E*t^3/(360-360*poisson^2)/a/b*(-3*b+3*poisson*b-30*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(-3*a-12*poisson*a+15*b^2/a),
E*t^3/(360-360*poisson^2)/a/b*(21-6*poisson+30*b^2/a^2+30*a^2/b^2),
E*t^3/(360-360*poisson^2)/a/b*(-3*b-12*poisson*b-30*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(3*a+12*poisson*a+30*b^2/a),
E*t^3/(360-360*poisson^2)/a/b*(-21+6*poisson-30*b^2/a^2+15*a^2/b^2),
E*t^3/(360-360*poisson^2)/a/b*(3*b+12*poisson*b-15*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(3*a-3*poisson*a+30*b^2/a);
E*t^3/(360-360*poisson^2)/a/b*(-3*b+3*poisson*b+15*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(2*b^2-2*poisson*b^2+10*a^2),
0,
E*t^3/(360-360*poisson^2)/a/b*(3*b-3*poisson*b+30*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(-2*b^2+2*poisson*b^2+20*a^2),
0,
```

```
E*t^3/(360-360*poisson^2)/a/b*(-3*b-12*poisson*b-30*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(8*b^2-8*poisson*b^2+40*a^2),
-30*E*t^3/(360-360*poisson^2)*poisson,
E*t^3/(360-360*poisson^2)/a/b*(3*b+12*poisson*b-15*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(-8*b^2+8*poisson*b^2+20*a^2),
0;
E*t^3/(360-360*poisson^2)/a/b*(3*a-3*poisson*a-15*b^2/a),
0,
E*t^3/(360-360*poisson^2)/a/b*(2*a^2-2*poisson*a^2+10*b^2),
E*t^3/(360-360*poisson^2)/a/b*(-3*a-12*poisson*a+15*b^2/a),
0,
E*t^3/(360-360*poisson^2)/a/b*(-8*a^2+8*poisson*a^2+20*b^2),
E*t^3/(360-360*poisson^2)/a/b*(3*a+12*poisson*a+30*b^2/a),
-30*E*t^3/(360-360*poisson^2)*poisson,
E*t^3/(360-360*poisson^2)/a/b*(8*a^2-8*poisson*a^2+40*b^2),
E*t^3/(360-360*poisson^2)/a/b*(-3*a+3*poisson*a-30*b^2/a),
0,
E*t^3/(360-360*poisson^2)/a/b*(-2*a^2+2*poisson*a^2+20*b^2);
E*t^3/(360-360*poisson^2)/a/b*(-21+6*poisson+15*b^2/a^2-30*a^2/b^2),
E*t^3/(360-360*poisson^2)/a/b*(-3*b+3*poisson*b-30*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(3*a+12*poisson*a-15*b^2/a),
E*t^3/(360-360*poisson^2)/a/b*(21-6*poisson-15*b^2/a^2-15*a^2/b^2),
E*t^3/(360-360*poisson^2)/a/b*(3*b-3*poisson*b-15*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(3*a-3*poisson*a-15*b^2/a),
E*t^3/(360-360*poisson^2)/a/b*(-21+6*poisson-30*b^2/a^2+15*a^2/b^2),
E*t^3/(360-360*poisson^2)/a/b*(3*b+12*poisson*b-15*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(-3*a+3*poisson*a-30*b^2/a),
E*t^3/(360-360*poisson^2)/a/b*(21-6*poisson+30*b^2/a^2+30*a^2/b^2),
E*t^3/(360-360*poisson^2)/a/b*(-3*b-12*poisson*b-30*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(-3*a-12*poisson*a-30*b^2/a);
E*t^3/(360-360*poisson^2)/a/b*(3*b-3*poisson*b+30*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(-2*b^2+2*poisson*b^2+20*a^2),
0,
E*t^3/(360-360*poisson^2)/a/b*(-3*b+3*poisson*b+15*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(2*b^2-2*poisson*b^2+10*a^2),
0,
E*t^3/(360-360*poisson^2)/a/b*(3*b+12*poisson*b-15*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(-8*b^2+8*poisson*b^2+20*a^2),
0,
E*t^3/(360-360*poisson^2)/a/b*(-3*b-12*poisson*b-30*a^2/b),
E*t^3/(360-360*poisson^2)/a/b*(8*b^2-8*poisson*b^2+40*a^2),
30*E*t^3/(360-360*poisson^2)*poisson;
E*t^3/(360-360*poisson^2)/a/b*(3*a+12*poisson*a-15*b^2/a),
0,
E*t^3/(360-360*poisson^2)/a/b*(-8*a^2+8*poisson*a^2+20*b^2),
E*t^3/(360-360*poisson^2)/a/b*(-3*a+3*poisson*a+15*b^2/a),
0,
```

```
E*t^3/(360-360*poisson^2)/a/b*(2*a^2-2*poisson*a^2+10*b^2),
E*t^3/(360-360*poisson^2)/a/b*(3*a-3*poisson*a+30*b^2/a),
0,
E*t^3/(360-360*poisson^2)/a/b*(-2*a^2+2*poisson*a^2+20*b^2),
E*t^3/(360-360*poisson^2)/a/b*(-3*a-12*poisson*a-30*b^2/a),
30*E*t^3/(360-360*poisson^2)*poisson,
E*t^3/(360-360*poisson^2)/a/b*(8*a^2-8*poisson*a^2+40*b^2)];
w=a*b*t*density;
syms kx yt kxi yti real;
ni=1/8*(1+kx*kxi)*(1+yt*yti)*(2+kx*kxi+yt*yti-kx^2-yt^2);
nix=-1/8*b*yti*(1+kx*kxi)*(1+yt*yti)*(1-yt^2);
niy=1/8*a*kxi*(1+kx*kxi)*(1+yt*yti)*(1-kx^2);
n(1)=subs(ni,{kxi,yti},{-1,-1});
n(2)=subs(nix,{kxi,yti},{-1,-1});
n(3)=subs(niy,{kxi,yti},{-1,-1});

n(4)=subs(ni,{kxi,yti},{1,-1});
n(5)=subs(nix,{kxi,yti},{1,-1});
n(6)=subs(niy,{kxi,yti},{1,-1});

n(7)=subs(ni,{kxi,yti},{1,1});
n(8)=subs(nix,{kxi,yti},{1,1});
n(9)=subs(niy,{kxi,yti},{1,1});

n(10)=subs(ni,{kxi,yti},{-1,1});
n(11)=subs(nix,{kxi,yti},{-1,1});
n(12)=subs(niy,{kxi,yti},{-1,1});

temp=n'*n;
m1=int(temp,kx,-1,1);
m=int(m1,yt,-1,1);
m=m*w;
m=double(m);
```

分别使用上述 MATLAB 程序和 ANSYS 计算结果，并将其列举在表 7.1 中. 由于两者使用了不同的单元类型，所以计算结果稍有区别. 在第 9 阶频率后，出现了 ANSYS 计算结果中独有的模态(在表中已经除去)，这是由于两种单元节点自由度不同的缘故.

表 7.1 一边固支方板的频率

模态阶数	MATLAB 计算结果(Hz)	ANSYS(shell63 单元)计算结果(Hz)
1	11.2078	11.211
2	27.4749	27.547
3	68.8768	68.888
4	87.7153	87.889

续表

模态阶数	MATLAB 计算结果(Hz)	ANSYS(shell63 单元)计算结果(Hz)
5	99.9788	100.40
6	174.2138	175.80
7	198.2866	197.93
8	207.2652	207.68
9	229.2561	230.16
10	296.9568	300.13
11	310.6423	314.65
12	386.5587	385.50
13	401.7931	401.37
14	416.9259	418.14
15	441.7086	450.04

7.9.2　计算 7.7 节中介绍壳体单元的单元刚度矩阵

同前几章介绍过的各种单元一样，计算 7.7 节中壳体单元的单元刚度矩阵需要完成式(7-179)的积分. 与前几章介绍的单元不同的是，计算式(7-179)中 $\boldsymbol{B}$ 矩阵的过程中需要求解 $\boldsymbol{J}^{-1}$，即雅可比矩阵的逆. 在这里，虽然雅可比矩阵可以通过符号运算推导出来，但是其表达式已经相当复杂，再通过符号运算求其逆是不可行的. 因此只有采用数值方法，在高斯积分的各积分点上逐个求出相应的数值型雅可比矩阵，然后才可以方便地求得其逆矩阵.

在下面的程序中，采用了在 ξ, η 和 ζ 方向都是两个积分点的缩减积分方案.

```
function [ek]=shellek(dyhm,jdzb,jdzb1,dybh)
%----------------------------------------------------------------
% dyhm: node connectivity for each element
% jdzb: coordinate of the nodes in the top surface of element
% jdzb1: coordinate of the nodes in the bottom surface of element
% dybh: the number of element whose stiffness matrix is calculated in
% this time of invoking.
temp=0.577350292; % value needed by Gauss integral
gaosi=[ 1  1  1;
        1 -1  1;
       -1  1  1;
       -1 -1  1;
        1  1 -1;
        1 -1 -1;
       -1  1 -1;
       -1 -1 -1];
d=[…]; % elastic matrix in 7.7-25. five order square matrix.
```

```
% Its value is determined by the character of material and isn't given
% in this example.

ek=zeros(40);
    for i=1:8
        for j=1:3
            v3i(i,j)=jdzb1(dybh(dyhm,i),j)-jdzb(dybh(dyhm,i),j);
        end
    end
    tempi=[1 0 0];
    for i=1:8
       temp1=cross(v3i(i,:)',tempi);
       xv2i(i,:)=temp1/norm(temp1);
       temp1=cross(xv2i(i,:),v3i(i,:));
       xv1i(i,:)=temp1/norm(temp1);
    end
    xv3i=v3i/t;

    for gaosii=1:8  % begin of Gauss integral
    zb1=temp*gaosi(gaosii,1);
    zb2=temp*gaosi(gaosii,2);
    zb3=temp*gaosi(gaosii,3);

    ni(1)=1/4*(1+zb1)*(1+zb2)*(zb1+zb2-1);
    ni(2)=1/4*(1-zb1)*(1+zb2)*(-zb1+zb2-1);
    ni(3)=1/4*(1-zb1)*(1-zb2)*(-zb1-zb2-1);
    ni(4)=1/4*(1+zb1)*(1-zb2)*(zb1-zb2-1);
    ni(5)=1/2*(1-zb1^2)*(1+zb2);
    ni(6)=1/2*(1-zb1)*(1-zb2^2);
    ni(7)=1/2*(1-zb1^2)*(1-zb2);
    ni(8)=1/2*(1+zb1)*(1-zb2^2);

v3=ni(1)*v3i(1,:)'+ni(2)*v3i(2,:)'+ni(3)*v3i(3,:)'+ni(4)*v3i(4,:)'+ni(5)
*v3i(5,:)'+ni(6)*v3i(6,:)'+ni(7)*v3i(7,:)'+ni(8)*v3i(8,:)';
    temp1=cross(v3,tempi');
    tn2=sqrt(temp1(1)^2+temp1(2)^2+temp1(3)^2);
    xv2=temp1/tn2;
    temp1=cross(xv2,v3);
    tn1=sqrt(temp1(1)^2+temp1(2)^2+temp1(3)^2);
    xv1=temp1/tn1;

    tn3=sqrt(v3(1)^2+v3(2)^2+v3(3)^2);
    xv3=v3/tn3;
    theta=[xv1,xv2,xv3];

    for i=1:8
       for j=1:3
```

```
    zmtemp(i,j)=(jdzb1(dybh(dyhm,i),j)+jdzb(dybh(dyhm,i),j))/2;
  end
end


pni(1,1)=1/4*(1+zb2)*(zb1+zb2-1)+(1/4+1/4*zb1)*(1+zb2);
pni(1,2)=(1/4+1/4*zb1)*(zb1+zb2-1)+(1/4+1/4*zb1)*(1+zb2);
pni(1,3)=0;
pni(2,1)=-1/4*(1+zb2)*(-zb1+zb2-1)-(1/4-1/4*zb1)*(1+zb2);
pni(2,2)=(1/4-1/4*zb1)*(-zb1+zb2-1)+(1/4-1/4*zb1)*(1+zb2);
pni(2,3)=0;
pni(3,1)=-1/4*(1-zb2)*(-zb1-zb2-1)-(1/4-1/4*zb1)*(1-zb2);
pni(3,2)=-(1/4-1/4*zb1)*(-zb1-zb2-1)-(1/4-1/4*zb1)*(1-zb2);
pni(3,3)=0;
pni(4,1)=1/4*(1-zb2)*(zb1-zb2-1)+(1/4+1/4*zb1)*(1-zb2);
pni(4,2)=-(1/4+1/4*zb1)*(zb1-zb2-1)-(1/4+1/4*zb1)*(1-zb2);
pni(4,3)=0;
pni(5,1)=-zb1*(1+zb2);
pni(5,2)=1/2-1/2*zb1^2;
pni(5,3)=0;
pni(6,1)=-1/2+1/2*zb2^2;
pni(6,2)=-2*(1/2-1/2*zb1)*zb2;
pni(6,3)=0;
pni(7,1)=-zb1*(1-zb2);
pni(7,2)=-1/2+1/2*zb1^2;
pni(7,3)=0;
pni(8,1)=1/2-1/2*zb2^2;
pni(8,2)=-2*(1/2+1/2*zb1)*zb2;
pni(8,3)=0;

x1=zmtemp(1,1);
y1=zmtemp(1,2);
z1=zmtemp(1,3);
x2=zmtemp(2,1);
y2=zmtemp(2,2);
z2=zmtemp(2,3);
x3=zmtemp(3,1);
y3=zmtemp(3,2);
z3=zmtemp(3,3);
x4=zmtemp(4,1);
y4=zmtemp(4,2);
z4=zmtemp(4,3);
x5=zmtemp(5,1);
y5=zmtemp(5,2);
z5=zmtemp(5,3);
x6=zmtemp(6,1);
y6=zmtemp(6,2);
```

```
    z6=zmtemp(6,3);
    x7=zmtemp(7,1);
    y7=zmtemp(7,2);
    z7-zmtemp(7,3);
    x8 zmtemp(8,1);
    y8 zmtemp(8,2);
    z8=zmtemp(8,3);

    dx1=v3i(1,1);
    dy1=v3i(1,2);
    dz1=v3i(1,3);
    dx2=v3i(2,1);
    dy2=v3i(2,2);
    dz2=v3i(2,3);
    dx3=v3i(3,1);
    dy3=v3i(3,2);
    dz3=v3i(3,3);
    dx4=v3i(4,1);
    dy4=v3i(4,2);
    dz4=v3i(4,3);
    dx5=v3i(5,1);
    dy5=v3i(5,2);
    dz5=v3i(5,3);
    dx6=v3i(6,1);
    dy6=v3i(6,2);
    dz6=v3i(6,3);
    dx7=v3i(7,1);
    dy7=v3i(7,2);
    dz7=v3i(7,3);
    dx8=v3i(8,1);
    dy8=v3i(8,2);
    dz8=v3i(8,3);

jtemp(1,1)=1/4*(1+zb2)*(zb1+zb2-1)*x1+(1/4+1/4*zb1)*(1+zb2)*x1-
1/4*(1+zb2)*(-zb1+zb2-1)*x2-(1/4-1/4*zb1)*(1+zb2)*x2-1/4*(1-zb2)*(-zb1-
zb2-1)*x3-(1/4-1/4*zb1)*(1-zb2)*x3+1/4*(1-zb2)*(zb1-zb2-
1)*x4+(1/4+1/4*zb1)*(1-zb2)*x4-zb1*(1+zb2)*x5-1/2*(1-zb2^2)*x6-zb1*(1-
zb2)*x7+1/2*(1-zb2^2)*x8+1/8*zb3*(1+zb2)*(zb1+zb2-
1)*dx1+1/2*zb3*(1/4+1/4*zb1)*(1+zb2)*dx1 1/8*zb3*(1+zb2)*(-zb1+zb2-
1)*dx2 1/2*zb3*(1/4 1/4*zb1)*(1+zb2)*dx2-1/8*zb3*(1-zb2)*(-zb1-zb2
1)*dx3-1/2*zb3*(1/4-1/4*zb1)*(1-zb2)*dx3+1/8*zb3*(1-zb2)*(zb1-zb2-
1)*dx4+1/2*zb3*(1/4+1/4*zb1)*(1-zb2)*dx4-1/2*zb3*zb1*(1+zb2)*dx5-
1/4*zb3*(1-zb2^2)*dx6-1/2*zb3*zb1*(1-zb2)*dx7+1/4*zb3*(1 zb2^2)*dx8;

jtemp(1,2)=1/4*(1+zb2)*(zb1+zb2-1)*y1+(1/4+1/4*zb1)*(1+zb2)*y1-
1/4*(1+zb2)*(-zb1+zb2-1)*y2-(1/4-1/4*zb1)*(1+zb2)*y2-1/4*(1-zb2)*(-zb1-
zb2-1)*y3-(1/4-1/4*zb1)*(1-zb2)*y3+1/4*(1-zb2)*(zb1-zb2-
```

```
1)*y4+(1/4+1/4*zb1)*(1-zb2)*y4-zb1*(1+zb2)*y5-1/2*(1-zb2^2)*y6-zb1*(1-
zb2)*y7+1/2*(1-zb2^2)*y8+1/8*zb3*(1+zb2)*(zb1+zb2-
1)*dy1+1/2*zb3*(1/4+1/4*zb1)*(1+zb2)*dy1-1/8*zb3*(1+zb2)*(-zb1+zb2-
1)*dy2-1/2*zb3*(1/4-1/4*zb1)*(1+zb2)*dy2-1/8*zb3*(1-zb2)*(-zb1-zb2-
1)*dy3-1/2*zb3*(1/4-1/4*zb1)*(1-zb2)*dy3+1/8*zb3*(1-zb2)*(zb1-zb2-
1)*dy4+1/2*zb3*(1/4+1/4*zb1)*(1-zb2)*dy4-1/2*zb3*zb1*(1+zb2)*dy5-
1/4*zb3*(1-zb2^2)*dy6-1/2*zb3*zb1*(1-zb2)*dy7+1/4*zb3*(1-zb2^2)*dy8;

jtemp(1,3)=1/4*(1+zb2)*(zb1+zb2-1)*z1+(1/4+1/4*zb1)*(1+zb2)*z1-
1/4*(1+zb2)*(-zb1+zb2-1)*z2-(1/4-1/4*zb1)*(1+zb2)*z2-1/4*(1-zb2)*(-zb1-
zb2-1)*z3-(1/4-1/4*zb1)*(1-zb2)*z3+1/4*(1-zb2)*(zb1-zb2-
1)*z4+(1/4+1/4*zb1)*(1-zb2)*z4-zb1*(1+zb2)*z5-1/2*(1-zb2^2)*z6-zb1*(1-
zb2)*z7+1/2*(1-zb2^2)*z8+1/8*zb3*(1+zb2)*(zb1+zb2-
1)*dz1+1/2*zb3*(1/4+1/4*zb1)*(1+zb2)*dz1-1/8*zb3*(1+zb2)*(-zb1+zb2-
1)*dz2-1/2*zb3*(1/4-1/4*zb1)*(1+zb2)*dz2-1/8*zb3*(1-zb2)*(-zb1-zb2-
1)*dz3-1/2*zb3*(1/4-1/4*zb1)*(1-zb2)*dz3+1/8*zb3*(1-zb2)*(zb1-zb2-
1)*dz4+1/2*zb3*(1/4+1/4*zb1)*(1-zb2)*dz4-1/2*zb3*zb1*(1+zb2)*dz5-
1/4*zb3*(1-zb2^2)*dz6-1/2*zb3*zb1*(1-zb2)*dz7+1/4*zb3*(1-zb2^2)*dz8;

jtemp(2,1)=(1/4+1/4*zb1)*(zb1+zb2-1)*x1+(1/4+1/4*zb1)*(1+zb2)*x1+(1/4-
1/4*zb1)*(-zb1+zb2-1)*x2+(1/4-1/4*zb1)*(1+zb2)*x2-(1/4-1/4*zb1)*(-zb1-
zb2-1)*x3-(1/4-1/4*zb1)*(1-zb2)*x3-(1/4+1/4*zb1)*(zb1-zb2-1)*x4-
(1/4+1/4*zb1)*(1-zb2)*x4+(1/2-1/2*zb1^2)*x5-2*(1/2-1/2*zb1)*zb2*x6-(1/2-
1/2*zb1^2)*x7-2*(1/2+1/2*zb1)*zb2*x8+1/2*zb3*(1/4+1/4*zb1)*(zb1+zb2-
1)*dx1+1/2*zb3*(1/4+1/4*zb1)*(1+zb2)*dx1+1/2*zb3*(1/4-1/4*zb1)*(-
zb1+zb2-1)*dx2+1/2*zb3*(1/4-1/4*zb1)*(1+zb2)*dx2-1/2*zb3*(1/4-
1/4*zb1)*(-zb1-zb2-1)*dx3-1/2*zb3*(1/4-1/4*zb1)*(1-zb2)*dx3-
1/2*zb3*(1/4+1/4*zb1)*(zb1-zb2-1)*dx4-1/2*zb3*(1/4+1/4*zb1)*(1-
zb2)*dx4+1/2*zb3*(1/2-1/2*zb1^2)*dx5-zb3*(1/2-1/2*zb1)*zb2*dx6-
1/2*zb3*(1/2-1/2*zb1^2)*dx7-zb3*(1/2+1/2*zb1)*zb2*dx8;

jtemp(2,2)=(1/4+1/4*zb1)*(zb1+zb2-1)*y1+(1/4+1/4*zb1)*(1+zb2)*y1+(1/4-
1/4*zb1)*(-zb1+zb2-1)*y2+(1/4-1/4*zb1)*(1+zb2)*y2-(1/4-1/4*zb1)*(-zb1-
zb2-1)*y3-(1/4-1/4*zb1)*(1-zb2)*y3-(1/4+1/4*zb1)*(zb1-zb2-1)*y4-
(1/4+1/4*zb1)*(1-zb2)*y4+(1/2-1/2*zb1^2)*y5-2*(1/2-1/2*zb1)*zb2*y6-(1/2-
1/2*zb1^2)*y7-2*(1/2+1/2*zb1)*zb2*y8+1/2*zb3*(1/4+1/4*zb1)*(zb1+zb2-
1)*dy1+1/2*zb3*(1/4+1/4*zb1)*(1+zb2)*dy1+1/2*zb3*(1/4-1/4*zb1)*(-
zb1+zb2-1)*dy2+1/2*zb3*(1/4-1/4*zb1)*(1+zb2)*dy2-1/2*zb3*(1/4-
1/4*zb1)*(-zb1-zb2-1)*dy3-1/2*zb3*(1/4-1/4*zb1)*(1-zb2)*dy3-
1/2*zb3*(1/4+1/4*zb1)*(zb1-zb2-1)*dy4-1/2*zb3*(1/4+1/4*zb1)*(1-
zb2)*dy4+1/2*zb3*(1/2-1/2*zb1^2)*dy5-zb3*(1/2-1/2*zb1)*zb2*dy6-
1/2*zb3*(1/2-1/2*zb1^2)*dy7-zb3*(1/2+1/2*zb1)*zb2*dy8;

jtemp(2,3)=(1/4+1/4*zb1)*(zb1+zb2-1)*z1+(1/4+1/4*zb1)*(1+zb2)*z1+(1/4-
1/4*zb1)*(-zb1+zb2-1)*z2+(1/4-1/4*zb1)*(1+zb2)*z2-(1/4-1/4*zb1)*(-zb1-
zb2-1)*z3-(1/4-1/4*zb1)*(1-zb2)*z3-(1/4+1/4*zb1)*(zb1-zb2-1)*z4-
(1/4+1/4*zb1)*(1-zb2)*z4+(1/2-1/2*zb1^2)*z5-2*(1/2-1/2*zb1)*zb2*z6-(1/2-
```

```
1/2*zb1^2)*z7-2*(1/2+1/2*zb1)*zb2*z8+1/2*zb3*(1/4+1/4*zb1)*(zb1+zb2-
1)*dz1+1/2*zb3*(1/4+1/4*zb1)*(1+zb2)*dz1+1/2*zb3*(1/4-1/4*zb1)*(-
zb1+zb2-1)*dz2+1/2*zb3*(1/4-1/4*zb1)*(1+zb2)*dz2-1/2*zb3*(1/4-
1/4*zb1)*(-zb1-zb2-1)*dz3-1/2*zb3*(1/4-1/4*zb1)*(1-zb2)*dz3-
1/2*zb3*(1/4+1/4*zb1)*(zb1-zb2-1)*dz4-1/2*zb3*(1/4+1/4*zb1)*(1-
zb2)*dz4+1/2*zb3*(1/2-1/2*zb1^2)*dz5-zb3*(1/2-1/2*zb1)*zb2*dz6-
1/2*zb3*(1/2-1/2*zb1^2)*dz7-zb3*(1/2+1/2*zb1)*zb2*dz8;

jtemp(3,1)=1/2*(1/4+1/4*zb1)*(1+zb2)*(zb1+zb2-1)*dx1+1/2*(1/4-
1/4*zb1)*(1+zb2)*( zb1+zb2-1)*dx2+1/2*(1/4-1/4*zb1)*(1-zb2)*(-zb1-zb2
1)*dx3+1/2*(1/4+1/4*zb1)*(1-zb2)*(zb1-zb2-1)*dx4+1/2*(1/2-
1/2*zb1^2)*(1+zb2)*dx5+1/2*(1/2-1/2*zb1)*(1-zb2^2)*dx6+1/2*(1/2-
1/2*zb1^2)*(1-zb2)*dx7+1/2*(1/2+1/2*zb1)*(1-zb2^2)*dx8;

jtemp(3,2)=1/2*(1/4+1/4*zb1)*(1+zb2)*(zb1+zb2-1)*dy1+1/2*(1/4-
1/4*zb1)*(1+zb2)*(-zb1+zb2-1)*dy2+1/2*(1/4-1/4*zb1)*(1-zb2)*(-zb1-zb2-
1)*dy3+1/2*(1/4+1/4*zb1)*(1-zb2)*(zb1-zb2-1)*dy4+1/2*(1/2-
1/2*zb1^2)*(1+zb2)*dy5+1/2*(1/2-1/2*zb1)*(1-zb2^2)*dy6+1/2*(1/2-
1/2*zb1^2)*(1-zb2)*dy7+1/2*(1/2+1/2*zb1)*(1-zb2^2)*dy8;

jtemp(3,3)=1/2*(1/4+1/4*zb1)*(1+zb2)*(zb1+zb2-1)*dz1+1/2*(1/4-
1/4*zb1)*(1+zb2)*(-zb1+zb2-1)*dz2+1/2*(1/4-1/4*zb1)*(1-zb2)*(-zb1-zb2-
1)*dz3+1/2*(1/4+1/4*zb1)*(1-zb2)*(zb1-zb2-1)*dz4+1/2*(1/2-
1/2*zb1^2)*(1+zb2)*dz5+1/2*(1/2-1/2*zb1)*(1-zb2^2)*dz6+1/2*(1/2-
1/2*zb1^2)*(1-zb2)*dz7+1/2*(1/2+1/2*zb1)*(1-zb2^2)*dz8;
    invj=inv(jtemp);
    dni=invj*[pni(1,1) pni(2,1) pni(3,1) pni(4,1) pni(5,1) pni(6,1)
              pni(7,1) pni(8,1);
                  pni(1,2) pni(2,2) pni(3,2) pni(4,2) pni(5,2) pni(6,2)
                  pni(7,2) pni(8,2);
                  0 0 0 0 0 0 0 0];
    dmi=invj*[zb3*pni(1,1) zb3*pni(2,1) zb3*pni(3,1) zb3*pni(4,1)
    zb3*pni(5,1) zb3*pni(6,1) zb3*pni(7,1) zb3*pni(8,1);
            zb3*pni(1,2) zb3*pni(2,2) zb3*pni(3,2) zb3*pni(4,2)
            zb3*pni(5,2) zb3*pni(6,2) zb3*pni(7,2) zb3*pni(8,2);
            ni(1) ni(2) ni(3) ni(4) ni(5) ni(6) ni(7) ni(8)];

    for bi=1:8
        for s=1:3
           alpha(s)=theta(1,s)*dni(1,bi)+theta(2,s)*dni(2,bi)
                   +theta(3,s)*dni(3,bi);
           beta(s)=(theta(1,s)*dmi(1,bi)+theta(2,s)*dmi(2,bi)
                  +theta(3,s)*dmi(3,bi))*t/2;
           gama(s)=theta(1,s)*xv1i(bi,1)+theta(2,s)*xv1i(bi,2)
                  +theta(3,s)*xv1i(bi,3);
           lmd(s)=theta(1,s)*xv2i(bi,1)+theta(2,s)*xv2i(bi,2)
                 +theta(3,s)*xv2i(bi,3);
```

```
        end
        btemp(:,:,bi)=[theta(1,1)*alpha(1) theta(2,1)*alpha(1)
        theta(3,1)*alpha(1) beta(1)*gama(1) beta(1)*lmd(1);
                     theta(1,2)*alpha(2) theta(2,2)*alpha(2)
                     theta(3,2)*alpha(2) beta(2)*gama(2) beta(2)*lmd(2);
                     theta(1,1)*alpha(2)+theta(1,2)*alpha(1)
                     theta(2,1)*alpha(2)+theta(2,2)*alpha(1)
                     theta(3,1)*alpha(2)+theta(3,2)*alpha(1)
                     beta(1)*gama(2)+beta(2)*gama(1)
                     beta(1)*lmd(2)+beta(2)*lmd(1);
                     theta(1,2)*alpha(3)+theta(1,3)*alpha(2)
                     theta(2,2)*alpha(3)+theta(2,3)*alpha(2)
                     theta(3,2)*alpha(3)+theta(3,3)*alpha(2)
                     beta(2)*gama(3)+beta(3)*gama(2)
                     beta(2)*lmd(3)+beta(3)*lmd(2);
                     theta(1,3)*alpha(1)+theta(1,1)*alpha(3)
                     theta(2,3)*alpha(1)+theta(2,1)*alpha(3)
                     theta(3,3)*alpha(1)+theta(3,1)*alpha(3)
                     beta(3)*gama(1)+beta(1)*gama(3)
                     beta(3)*lmd(1)+beta(1)*lmd(3)];
    end
    b=[btemp(:,:,1) btemp(:,:,2) btemp(:,:,3) btemp(:,:,4) btemp(:,:,5)
    btemp(:,:,6) btemp(:,:,7) btemp(:,:,8)];
    djt=det(jtemp);
    tk=b'*d*b*djt;
    ek=ek+tk;
end %end of Gauss integral
ek=tril(ek)+tril(ek)'-diag(diag(ek,0));
```

第8章 工程应用

8.1 结构动力学优化设计

有关统计表明，在飞行器所发生的重大事故中，40%与振动有关. 我国多个型号的飞机与航天器在研制过程中也均遇到过许多严重的振动问题. 对飞机而言，着陆、滑跑和起飞过程的地面载荷、大气紊流引起的突发载荷、湍流气动载荷，发动机噪声激励以及气流分离造成的抖振载荷都是动载荷. 所以说，在一定程度上，飞机性能与服役寿命依赖于飞机结构的动力学性能. 不言而喻，为了改善此类结构和系统的动态特性，以保证它们正常、可靠地工作，必须进行结构动力学设计. 事实上，在动力学优化设计问题提出之前，结构在传统上都是由先验知识或静力问题优化确定，然后再用动力学试验或验算对其修改，这其实是一种落后、费时且欠优的设计方法. 结构优化的目的在于以最少的材料、最低的造价和最简单的工艺来实现结构性能最佳. 结构优化最初采用经典解析方法求解，所用方法是变分法或微分法. 对于无约束优化问题，利用 Euler-Lagrange 方程构造极值存在的充要条件，然后用梯度向量搜索优化方向；约束优化问题则采用 Lagrange 乘子构成辅助函数来考虑约束条件的影响. 虽然解析方法可以解决一些简单构件，如桁架的优化问题，但涉及复杂的数学推导阻碍了它在实际结构中的应用. 近年来，计算机技术及其在结构分析中的普遍应用，促进了结构优化数值方法的发展. 其中，数学规划法(MP 法)和优化准则法(OC 法)是被广泛采用的两种. 近年来，仿生算法也逐渐得到了应用和发展. 显然，在一定的激励环境下，结构尺寸、形状、材料与拓扑构形控制着结构的振动响应水平. 因此，在给定动力学约束条件下，进行结构动力学优化设计，“主动”地确定结构最优的动力特性(也可同时满足静力问题的多种约束)成为结构优化设计领域近年来的一个活跃的研究分支.

结构优化设计可以根据设计变量的类型划分为不同的层次：在给定结构的类型、材料、布局拓扑和外形几何的情况下，优化各个组成构件的截面尺寸或截面性质(弹性模量)，使结构性能最佳，通常称为尺寸优化，它是结构优化设计中的最低层次. 如果让结构的几何外形也可以变化，例如，把桁架和刚架的节点位置或连续体边界形状的几何参数作为设计变量，优化又进入了一个较高的层次，即所谓的结构形状优化. 进而再允许对桁架节点联结关系或连续体结构的布局进行优化，则优化达到更高的层次，即结构的拓扑优化. 显然，随着结构优化层次的提高，其难度也越来越大，然而目标(如重量)上的优化效益亦越大.

8.1.1 优化问题基本描述

通常，结构动力学优化设计问题可表述为

$$
\text{minimize (or) maximize} \quad f(x)
$$

$$
\text{s.t} \quad \begin{cases} h_i(x)=0 & i=1,2,\cdots,p \\ g_j(x)\leqslant 0 & j=1,2,\cdots,q \end{cases} \tag{8-1}
$$

式中：$f(x)$ 为目标函数；$h_i(x)$ 和 $g_j(x)$ 为第 i 个等式约束函数和第 j 个不等式约束函数；p 和 q 为其相应约束函数的数目，x 是设计变量矢量.

8.1.2 动力学尺寸优化

1. 空间桁架固频极值的求解

1) 求解算法

极大值的求解过程由以下步骤完成

$$
\begin{aligned} &\max \quad \lambda_i = \lambda_i(A) \\ &\text{s.t} \quad A_t \geqslant 0 \qquad (t=1,2,\cdots,N_m) \end{aligned} \tag{8-2}
$$

式中：A_t 为结构设计变量；N_m 为可优化的设计变量的个数.

用最大下降速率法(SDM)，由初始点 A_t^0 开始反复迭代，并由下列规则使 λ_i 达到极大点

$$
A_t = A_t^0 + \alpha B_t \tag{8-3}
$$

其中

$$
B_t = \partial \lambda_i / \partial A_t \tag{8-4}
$$

α 是在 $\boldsymbol{B}=\left(B_1,B_2,\cdots,B_{N_m}\right)$ 方向上的迭代步长，可由下式得到

$$
B_t = \frac{\boldsymbol{\Phi}_i^{\mathrm{T}}\left(\dfrac{\partial \boldsymbol{K}}{\partial A_t} - \lambda_i \dfrac{\partial \boldsymbol{M}}{\partial A_t}\right)\boldsymbol{\Phi}_i}{\boldsymbol{\Phi}_i^{\mathrm{T}}\boldsymbol{M}\boldsymbol{\Phi}_i} \tag{8-5}
$$

式中，$\boldsymbol{\Phi}_i$ 是第 i 个特征向量. 若 $\boldsymbol{\Phi}_i$ 满足 $\boldsymbol{\Phi}_i^{\mathrm{T}}\boldsymbol{M}\boldsymbol{\Phi}_j=\delta_{ij}$，则式(8-5)变为

$$
B_t = -\boldsymbol{\Phi}_i^{\mathrm{T}}\left[\frac{\partial \boldsymbol{K}}{\partial A_t} - \lambda_i \frac{\partial \boldsymbol{M}}{\partial A_t}\right]\boldsymbol{\Phi}_i \tag{8-6}
$$

对于传统 SDM，α 是由最小 $\lambda_i\left(A_t+\alpha\bar{B}_t\right)$ $\alpha\geqslant 0$ 的线性搜索问题的解来确定的. 由于目标是固有频率，必须进行模态分析. 为省机时,将式(8-3)改为

$$
A_t = A_t^0 + \alpha \bar{B}_t \qquad (t=1,2,\cdots,N_m) \tag{8-7}
$$

其中

$$
\bar{B}_t = B_t \Big/ \sqrt{B_1^2 + B_2^2 + \cdots + B_t^2} \tag{8-8}
$$

$$
\alpha = c_0 / \alpha_0 \tag{8-9}
$$

$$
c_0 = \max\left(\frac{B_t}{\bar{B}_t}\right) \qquad 当 B_t < 0, \quad t=1,2,\cdots,N_m \tag{8-10}
$$

式中，α_0 为满足 $\alpha_0>1$ 的一个正权系数，由迭代过程确定. 式(8-8)用来正则化优化方向，而式(8-9)和式(8-10)分别用来确定步长大小和确保结构修改实现.

算法迭代终止应满足的所有条件为

$$\left|\lambda_i^{s+1}-\lambda_i^s\right|\leqslant\varepsilon_1 \tag{8-11}$$

$$\left|\frac{\partial\lambda_i}{\partial A_t}\right|\leqslant\varepsilon_2 \qquad (t=1,2,\cdots,N_m) \tag{8-12}$$

$$\left|A_t^{s+1}-A_t^s\right|\leqslant\varepsilon_3 \tag{8-13}$$

若式(8-11)与式(8-12)得到满足，则相应固频达到其最大值；而式(8-13)得到满足，则可以找到设计变量的一个设计范围，在此范围内，相应固频达到极大值. ε_1、ε_2和ε_3是事先由工程精度要求确定的误差控制量. 对于极小值的求解过程，式(8-6)相应变为：

$$B_t=\boldsymbol{\Phi}_i^{\mathrm{T}}\left[\frac{\partial\boldsymbol{K}}{\partial A_t}-\lambda_i\frac{\partial\boldsymbol{M}}{\partial A_t}\right]\boldsymbol{\Phi}_i \tag{8-14}$$

2) 数值算例

【例 8.1】 如图 8.1 所示的平面二杆桁架，弹性模量 $E=2.1\times10^{11}$Pa，质量密度 $\rho=7.8\times10^3$kg/m^3. 杆①和杆②的初始横截面积分别为5cm^{2}和8cm^2. 利用所提出的算法求解第一、第二阶特征值的极大值的过程分别如图 8.2 和图 8.3 所示.

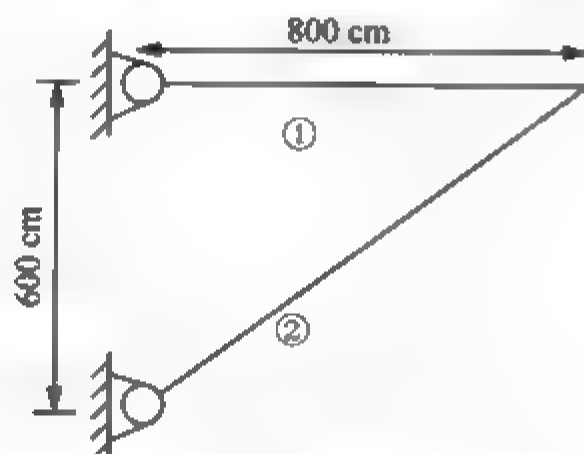


图 8.1 平面二杆桁架

```
%-----------------------------------------------------------------------%
Example 8.1
% to seek the maximal value of the first natural frequency for plane 2-bar truss
%-------------------------------------------------------------------------
 A0=[5 8];                              % initial cross-section area
v=ones(2,1);
k=1;
flag=1;
D(1)=0;
B0=5;
while flag
   m=3.12*A0(1)+3.9*A0(2);
M=[m 0;0 m];
k1=375*A0(1)+192*A0(2);
k2=144*A0(2);
k3=108*A0(2);
```

```
K=7.0e5*[k1 k2;k2 k3];
[V,D1]=eig(K,M);            % compute eigenvalue and eigenvector
[lambda,k4]=sort(diag(real((D1))));
V=real(V(:,k4));
Factor=diag(V'*M*V);
Vnorm=V*inv(sqrt(diag(Factor)));                  % normalize eigenvector
freq=sqrt(lambda)/(2*pi);
A0=[1  1];
for i=1:2
m=3.12*A0(1)+3.9*A0(2);
M=[m 0;0 m];
k1=375*A0(1)+192*A0(2);
k2=144*A0(2);
k3=108*A0(2);
K=7.0e5*[k1 k2;k2 k3];
v(i)=Vnorm(:,1)'*(K-lambda(1)*M)*Vnorm(:,1);     % sensitivity analysis
end
   sum=0.0;
   for i=1:2
      sum=sum+v(i)^2;
   end
   sum=sqrt(sum);
   for i=1:2
      v(i)=-v(i)/sum;
      n=0;
      if v(i)<0
         n=n+1;
         c0(n)=A0(i)/v(i);
     end
   end

c0max=max(c0);
B=c0max/B0;

for i=1:2
   A(i)=A0(i)+B*v(i);
end
A0=A;
v0=v;
k=k+1;
D(k)=lambda(1)
epilson=(abs(D(k)-D(k-1)))/D(k);

if epilson<=1e-5
   flag=0;
else
   flag=1;
```

```
    end
end

plot(D(2:k),'--*')
xlabel(' 迭代次数')
ylabel('第一特征值(/s^2)')
```

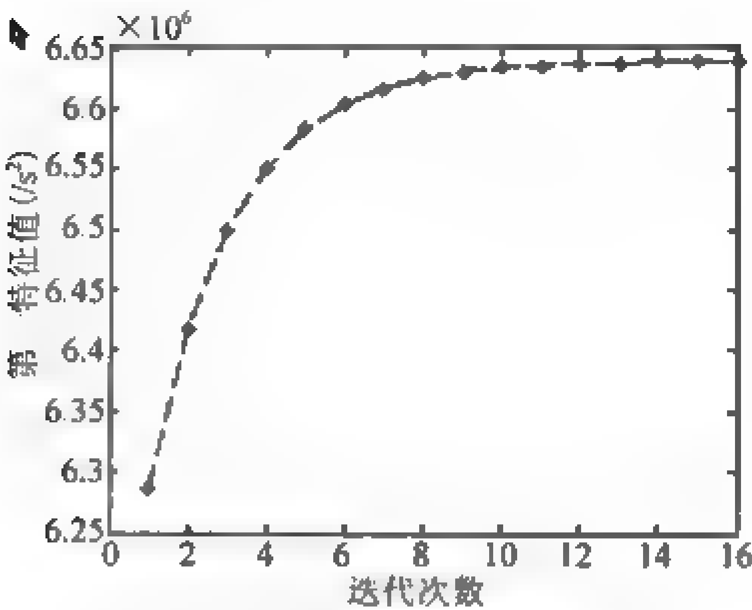


图 8.2　第一阶频率极大值的迭代过程

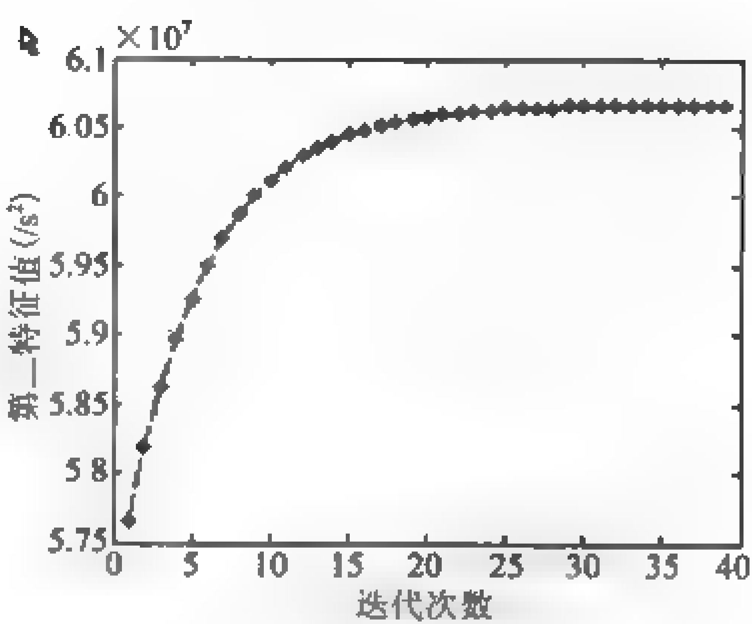


图 8.3　第二阶频率极大值的迭代过程

2. 桁架动力学尺寸优化

【例 8.2】 数值算例. 如图 8.4 所示的空间 72 杆桁架，弹性模量 $E=2.1\times10^{11}$Pa，质量密度 $\rho=7.8\times10^{3}$kg/m^3. 在节点 20 的 x 方向施加一大小为 50 000N 的力. 约束条件为：结构的基频约束要求基频不小于 100Hz，应力约束要求每个单元的应力不大于 173.2MPa，静位移约束为节点 20 的 x 方向的位移小于 0.001m. 优化的目的是在满足约束的前提下最小结构的质量. 普通桁架单元的横截面积的取值域为 $\boldsymbol{S}=\{0.774, 1.355, 2.142, 3.348, 4.632, 6.542, 7.742, 9.032, 10.839, 12.671, 14.581, 21.483, 34.839, 44.516, 52.903, 60.258\}$cm^2. 10 次随机运行的优化结果见表 8.1.

表 8.1　空间 72 杆桁架优化结果

次　数	质量(kg)
1	467.9667
2	477.1802
3	477.0647
4	480.1065
5	459.0711
6	460.6526
7	488.9148

续表

次 数	质量(kg)
8	475.1115
9	477.4006
10	467.7001

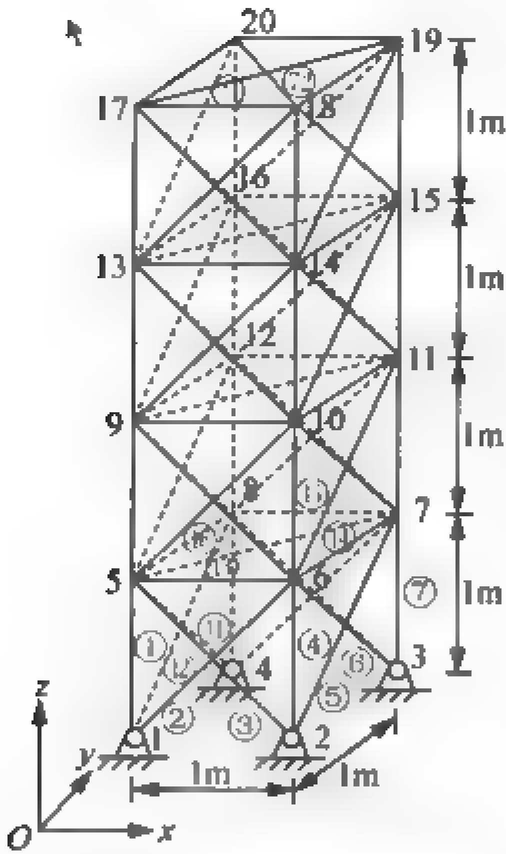


图8.4 空间72杆桁架

```
% ------- -  -   -----------------------  -----------% Ex 8.2
%  This program is used to optimize cross-section areas of
%  the space 72-bar truss based on genetic algorithm
% --------------  ---------------------  -------------- ------------
% Initialization of parameters
%     --------------------     -------------   -------------   --------
  LENGTH=4;                   %the chromosome length of design variable
  gan=72;                          %the number of gan
  CHROMLENGTH=LENGTH*gan';       %totle length of chromosome
  PopSize=400;                      %population size
  Pc=0.8;                          %probability of crossover
  Pm=0.02;                         %probability of mutation
  penal=100;                       %the initial penalty
%---- ---------          - -------         ------
% Definition of data structure
%-          ---------        -------          ----
 for loopi=1:PopSize
   for loopj=1:CHROMLENGTH
      populationchrom(loopi,loopj)=0.0;
```

```
      end
    end
    for loopi=1:3
      for loopj=1:CHROMLENGTH
          bestworstchrom(loopi,loopj)=0.0;
end
end
%----------  ------------- --------------------
% Begin the main program
% ------- ------------ ----------------------.--

populationchrom=GenerateInitialPopulation(PopSize,CHROMLENGTH,LENGTH,gan);
populationfitness=CalculateObjectValue(populationchrom,LENGTH,PopSize,gan,
penal);
bestworstchrom=FindBestAndWorstIndividual(populationchrom,populationfitness,
PopSize,bestworstchrom);
bestworstchrom(3,:)=bestworstchrom(1,:);
%-------the ceasing condition--------------
 itercri=1;
  generation=0;
   while itercri

populationchrom=GenerateNextPopulation(PopSize,populationchrom,CHROMLENG
TH,LENGTH,Pc,Pm,populationfitness);
 populationfitness=CalculateObjectValue(populationchrom,LENGTH,PopSize,
gan,penal); bestworstchrom=FindBestAndWorstIndividual(populationchrom,
populationfitness,PopSize,bestworstchrom);
%----output the result of current population----------
     for i=1:3
       area=Decode(bestworstchrom(i,:),LENGTH,gan);
        [W,dispmax,stressmax,frequency]=Trusssizeopt(area);
        C=VALUE(W,dispmax,stressmax,frequency,penal);
        bestworstfitness(i)=C;
    end

   if bestworstfitness(1) > bestworstfitness(3)
        bestworstchrom(3,:)=bestworstchrom(1,:);
   end
     area=Decode(bestworstchrom(3,:),LENGTH,gan);
     [W,dispmax,stressmax,frequency]=Trusssizeopt(area)
     sumfitness=0.0;
     for i=1:PopSize
        sumfitness=sumfitness+populationfitness(i);
     end
     meanfitness=(sumfitness-min(populationfitness)+bestworstfitness(3))/PopSize;
```

```
     generation=generation+1
     maxfitdata(generation)=bestworstfitness(3);
     meanfitdata(generation)=meanfitness;
     Edata(generation)=W;
    if generation>50
        h1=abs(Edata(generation)-Edata(generation-50))/Edata(generation);
        if h1<1e-4& frequency>=80&stressmax<=173.2e6&dispmax<=1e-3
            itercri=0;
        else
            itercri=1;
        end
    end
    if penal>=2;
     penal=penal/2;
    end
  end
%----------------------------------------------------------------------
% The end
%----------------------------------------------------------------------

function populationchrom=GenerateInitialPopulation(PopSize,CHROMLENGTH,
LENGTH,gan)
%----------------------------------------------------------------------
% generate the first population
%----------------------------------------------------------------------

  for loopi=1:PopSize
      for loopj=1:(LENGTH*gan)
        if rand <=0.5
           populationchrom(loopi,loopj)=1;
        else
           populationchrom(loopi,loopj)=0;
        end
     end
  end

function
populationfitness=CalculateObjectValue(populationchrom,LENGTH,PopSize,gan,
penal)
%----------------------------------------------------------------------
% to calculate population fitness
%-------------------------------------------------

for loopi=1:PopSize
        area=Decode(populationchrom,LENGTH,gan);
        [W,dispmax,stressmax,frequency]=Trusssizeopt(area);
        Cv=VALUE(W,dispmax,stressmax,frequency,penal);
```

```
        populationfitness(loopi)=Cv;
    end


    function bestworstchrom=FindBestAndWorstIndividual(populationchrom,
    populationfitness,PopSize,bestworstchrom);
    %-------  -------  ------  --  ------  ------  ------  ------
    % to find out the best and worst individual so far current generation
    %  -----  -----  ------  ----  ------  --  ------  ------- --
      bestworstfitness(1)=populationfitness(1);
      bestworstfitness(2)=populationfitness(1);
       flag=1;
     while flag
          num=1;
       for loopi=2:PopSize
           num=num+1;
        if  populationfitness(loopi)>=bestworstfitness(1)
           bestworstchrom(1,:)=populationchrom(loopi,:);
           bestworstfitness(1)=populationfitness(loopi);
        end
        if  populationfitness(loopi)<=bestworstfitness(2)
           bestworstchrom(2,:)=populationchrom(loopi,:);
           bestworstfitness(2)=populationfitness(loopi);
        end
        if num==PopSize
            flag=0;
        end
      end
    end

    function populationchrom=GenerateNextPopulation(PopSize,populationchrom,
    CHROMLENGTH,LENGTH,Pc,Pm,populationfitness)
    %------------------------------  -----  ----------------------------
    %  generate next population:select,crossover and mutate
    %---  ------------------------------  -----  -----  ----
    % to reproduce a chromosome by proportional selection
    %----  ------  -----  -----  -------------  ----  ---
      p=0.0;
      sum=0.0;
      %--   ------  -----  -----
      %  calculate relative fitness
      %---   ----   ----      -  -
      for i=1:PopSize
         sum=sum+populationfitness(i);
      end
      for i=1:PopSize
         cfitness(i)=populationfitness(i)/sum;
```

```
end
%-----------------------------
% calculate cumulative fitness
%-----------------------------
for i=2:PopSize
   cfitness(i)=cfitness(i-1)+cfitness(i);
end
%-----------------------------
% selection operation
%-----------------------------
for i=1:PopSize
   p=rand;
   index=1;
   while p > cfitness(index)
      index=index+1;
   end
   newpopulationchrom(i,:)=populationchrom(index,:);
end
for i=1:PopSize
   populationchrom(i,:)=newpopulationchrom(i,:);
end

%---------------------------------------------------------------
% crossover two chromosome by means of two-point crossover
%---------------------------------------------------------------
% make a pair of individual randomly
%------------------------------------
for i=1:PopSize
   index(i)=i;
end
for i=0:PopSize-1
   point=round(random('unif',0,PopSize-i));
   temp=index(i+1);
   if (point+i+1) > PopSize
      index(i+1)=index(point+i);
      index(point+i)=temp;
   else
      index(i+1)=index(point+i+1);
      index(point+i+1)=temp;
   end
end
%-------------------------------------
% two point crossover operation
%-------------------------------------
for i=1:2:PopSize
   p=rand;
   if p<Pc
```

```
        point(1)=round(random('unif',1,CHROMLENGTH));
        point(2)=round(random('unif',1,CHROMLENGTH));
        if point(1)==point(2)
          ch=populationchrom(index(i),1:point(1));
          populationchrom(index(i),1:point(1))=populationchrom(index(i+1),
          1:point(1));
          populationchrom(index(i+1),1:point(1))=ch;
        elseif point(1)>point(2)
          ch=populationchrom(index(i),point(2):point(1));
          populationchrom(index(i),point(2):point(1))=populationchrom(index(i+1),
          point(2):point(1));
          populationchrom(index(i+1),point(2):point(1))=ch;
        else
          ch=populationchrom(index(i),point(1):point(2));
          populationchrom(index(i),point(1):point(2))=populationchrom(index(i+1),
          point(1):point(2));
          populationchrom(index(i+1),point(1):point(2))=ch;
       end
     end
     end
  %------------------------------------------------------------
  %          mutation of a chromosome
  %------------------------------------------------------------
  for i=1:PopSize
     for j=1:CHROMLENGTH
        p=rand;
        if p<Pm
           if populationchrom(i,j)==0;
              populationchrom(i,j)=1;
           else
              populationchrom(i,j)=0;
           end
        end
     end
  end

function area=Decode(populationchrom,LENGTH,gan)
%------------------------------------------------------------
% decode
% ouput variables:
%  area =cross sectional areas of structural elements
%------------------------------------------------------------
% extract structural cross section areas
%------------------------------------------------------------
 vdarea=1e-6*[3125 80 100 125 180  259.2 320 405 500  672.2  793.8
1125  1280  1620  2000  2420];
 area=zeros(gan(1),1);
```

```
  for loopi=1:gan(1)
      temp=0;
     for loopj=1:LENGTH(1)
        if populationchrom((loopi-1)*LENGTH(1)+loopj)==1
           temp=temp+2^(loopj-1);
        end
     end
     temp=temp+1;
     area(loopi)=vdarea(temp);
  end

function [W,dispmax,stressmax,frequency]=Trusssizeopt(area)
%-----------------------------------------------------------------
% Purpose:
% This function is used to calculate total mass, maximal nodal diaplcement,
%  maximal element stress and first natural frequency for space 72-bar truss
%
% Synopsis:
%       [W,dispmax,stressmax,frequency]=Trusssizeopt(area)
% Variable Description:
%       Input parameters
%       area - element cross-section areas
%       Output parameters
%       W - total truss mass
%       dispmax - maximal nodal displacement
%       stressmax - maximal element stress
%       frequency - first natural frequency
%     ---------------------------------------------------------------
E=2.1e11;                              % elastic modulus
density=7860;                           % mass density
node_number=20;
element_number=72;
No_dof=3;
No_nel=2;
Sdof=node_number*No_dof;     % total number of dofs for ground structure
f=zeros(Sdof,1);                         % force vector
f(58)=-50000;
nc=[0 0 0;1 0 0;1 1 0;0 1 0; ...   % nodal coordinate for ground structure
    0 0 1;1 0 1;1 1 1;0 1 1; ...
    0 0 2;1 0 2;1 1 2;0 1 2; ...
     0 0 3;1 0 3;1 1 3;0 1 3; ...
     0 0 4;1 0 4;1 1 4;0 1 4];
en=zeros(72,2);
en(1,1)=1;en(1,2)=5;                                        %en:element_node
en(2,1)=1;en(2,2)=6;
en(3,1)=2;en(3,2)=5;
en(4,1)=2;en(4,2)=6;
```

```
en(5,1)=2;en(5,2)=7;
en(6,1)=3;en(6,2)=6;
en(7,1)=3;en(7,2)=7;
en(8,1)=3;en(8,2)=8;
en(9,1)=4;en(9,2)=7;
en(10,1)=4;en(10,2)=8;
en(11,1)=4;en(11,2)=5;
en(12,1)=1;en(12,2)=8;
en(13,1)=5;en(13,2)=6;
en(14,1)=6;en(14,2)=7;
en(15,1)=7;en(15,2)=8;
en(16,1)=5;en(16,2)=8;
en(17,1)=6;en(17,2)=8;
en(18,1)=5;en(18,2)=7;
for loopi=1:3
    for loopj=1:18
       en(loopi*18+loopj,1)=en(loopj,1)+4*loopi;
       en(loopi*18+loopj,2)=en(loopj,2)+4*loopi;
    end
end

ed(1:node_number,1:3)=1;                              %ed:element_displacement
constraint=[1,1;1,2;1,3;2,1;2,2;2,3;3,1;3,2;3,3;4,1;4,2;4,3];
for loopi=1:length(constraint);
    ed(constraint(loopi,1),constraint(loopi,2))=0;
end
dof=0;
for loopi=1:node_number
     for loopj=1:3
        if ed(loopi,loopj)~=0
            dof=dof+1;
            ed(loopi,loopj)=dof;
        end
    end
end
%-------------------------------------------------
% Initilization to zero
%-------------------------------------------------
k(1:dof,1:dof)=0;                  %structural_stiffness_matrix
m(1:dof,1:dof)=0;;                 %structural_mass_matrix
disp(1:dof)=0;                    % system displacement vector
eldisp(1:No_nel*No_dof)=0;          % element nodal displacement vector
elforce(1:No_nel*No_dof)=0;         % element force vector
stress(1:element_number)=0;          % stress vector for every element
el(1:element_number)=0;
e2s(1:6)=0;     % index of transform the element displament number to
structural
```

```
%------------------------------------------------------------------
% Assemble system mass and siffness matrix
%------------------------------------------------------------------
W=0;
for loopi=1:element_number
    for zi=1:2
        e2s((zi-1)*2+1)=ed(en(loopi,zi),1);
        e2s((zi-1)*2+2)=ed(en(loopi,zi),2);
        e2s((zi-1)*2+3)=ed(en(loopi,zi),3);
    end
el(loopi)=sqrt((nc(en(loopi,1),1)-nc(en(loopi,2),1))^2+(nc(en(loopi,1),2)
-nc(en(loopi,2),2))^2 ...
              +(nc(en(loopi,1),3)-nc(en(loopi,2),3))^2);
    c=(nc(en(loopi,1),1)-nc(en(loopi,2),1))/el(loopi);
    s=(nc(en(loopi,1),2)-nc(en(loopi,2),2))/el(loopi);
    r=(nc(en(loopi,1),3)-nc(en(loopi,2),3))/el(loopi);
    dk=E*area(loopi)/el(loopi)*[c^2 c*s c*r -c^2 -c*s -c*r; ...
    % element stiffness matrix
                                c*s s^2 s*r -c*s -s^2 -s*r; ...
                                c*r s*r r^2 -c*r -s*r -r^2; ...
                                -c^2 -c*s -c*r c^2 c*s c*r; ...
                                -c*s -s^2 -s*r c*s s^2 s*r; ...
                                -c*r -s*r -r^2 c*r s*r r^2];
    dm=(density*area(loopi)*el(loopi))/2*eye(6);    % element mass matrix
    for jx=1:6
        for jy=1:6
            if(e2s(jx)*e2s(jy)~=0)
              k(e2s(jx),e2s(jy))=k(e2s(jx),e2s(jy))+dk(jx,jy);
              m(e2s(jx),e2s(jy))=m(e2s(jx),e2s(jy))+dm(jx,jy);
            end
        end
    end
    W=W+density*area(loopi)*el(loopi);
end
%------------------------------------------------
% Compute the first natural frequency
%------------------------------------------------
p=1;                            % the number of natural frequencies
epsilon=1e-5;
[v,d]=Inviter(k,m,p,epsilon);    % Inverse iterative method
frequency=sqrt(d)/(2*pi);
%------------------------------------------------
% Compute system displacement
%------------------------------------------------
Number_con=-1;                  % apply boundary conditions
for loopi=1:length(constraint)
    Number_con= Number_con+1;
```

```
     f((constraint(loopi,1)-1)*No_dof+constraint(loopi,2)-Number_con)=[];
  end
  disp=k\f;                    % solve matrix equation for nodal
displacement
%------       ---------        --------
% post computation for stress calculation
%-------       ----   ----------    ------
  for loopi=1:element_number
    for zi=1:2
        e2s((zi-1)*2+1)=ed(en(loopi,zi),1);
        e2s((zi-1)*2+2)=ed(en(loopi,zi),2);
        e2s((zi-1)*2+3)=ed(en(loopi,zi),3);
    end
el(loopi)=sqrt((nc(en(loopi,1),1)-nc(en(loopi,2),1))^2+(nc(en(loopi,1),2)
-nc(en(loopi,2),2))^2 ...
              +(nc(en(loopi,1),3)-nc(en(loopi,2),3))^2);
    c=(nc(en(loopi,1),1)-nc(en(loopi,2),1))/el(loopi);
    s=(nc(en(loopi,1),2)-nc(en(loopi,2),2))/el(loopi);
    r=(nc(en(loopi,1),3)-nc(en(loopi,2),3))/el(loopi);
    dk=E*area(loopi)/el(loopi)*[c^2 c*s c*r -c^2 -c*s -c*r; ...
                              c*s s^2 s*r -c*s -s^2 -s*r; ...
                              c*r s*r r^2 -c*r -s*r -r^2; ...
                              -c^2 -c*s -c*r c^2 c*s c*r; ...
                              -c*s -s^2 -s*r c*s s^2 s*r; ...
                              -c*r -s*r -r^2 c*r s*r r^2];
    for loopj=1:(No_nel*No_dof)
       if e2s(loopj)==0
          eldisp(loopj)=0;
       else
          eldisp(loopj)=disp(e2s(loopj));
       end
    end
     elforce=dk*eldisp';                          % element force vector
    if area(loopi)==0
        stress(loopi)=0;
    else
       stress(loopi)=sqrt(elforce(1)^2+elforce(2)^2+elforce(3)^2)/area(loopi);
% stress
    end
    if ((nc(en(loopi,1),1)-nc(en(loopi,2),1))*elforce(4))<0
% check if tension or compression
       stress(loopi)=-stress(loopi);
    end
  end
%  -----    -----  - - --      --  -  --
% Construct the taotal displacement
%------   -----  -----    ----    ----
```

```
 flag-1;
 iter--1;
 displ=disp;
 while flag
     for loopi=1:length(constraint)
         iter=iter+1;
         for loopj=1:(dof+iter)
             if loopj<((constraint(loopi,1)-1)*No_dof+constraint(loopi,2))
                 dispt(loopj)=displ(loopj);
             else
                 dispt(loopj+1)=displ(loopj);
             end
         end
         dispt((constraint(loopi,1)-1)*No_dof+constraint(loopi,2))=0;
         displ=dispt;
     end
     if length(displ)>=node_number*No_dof
             flag=0;
     end
 end
%-----------------------------------------------------------
%  Output solutions
%-----------------------------------------------------------
  dispmax=max(abs(dispt'));
  stressmax=max(abs(stress'));
function Cv=VALUE(W,dispmax,stressmax,frequency,penal)
%--------------------------------------------------------------------------------
%  compute object value by penalty method
%--------------------------------------------------------------------------------
if   dispmax<=1e-3
      d1=0.0;
else
      d1=(dispmax-1e-3)/(1e-3);
end
if   stressmax<=1.732e8
      d2=0.0;
else
      d2=(stressmax-1.732e8)/(1.732e8);
end
if   frequency>=80
      d3=0.0;
else
      d3=(80-frequency)/80;
end
   d-d1+d2+d3;
   Cv=1/(W*(1+penal*d));
```

8.1.3　动力学拓扑优化

结构动力学拓扑优化是结构优化中最富有挑战性的研究领域. 结构拓扑的改进可大大改善结构的静、动态性能，并可使无解的问题变得有解. 目前，结构的动力学拓扑优化比起动力学截面优化和动力学形状优化来研究甚少，几乎刚刚起步. 针对离散结构的动力学拓扑优化设计，其优化方法主要是建立在基结构基础上. 由于连续体结构拓扑优化模型描述的困难和数值算法的巨大计算量，因而发展较慢. 目前的方法都是在基结构基础上的描述方式，包括几何(尺寸)描述方式和材料(物理)描述方式.

1. 桁架拓扑优化

【例 8.3】 数值算例. 如图 8.5 所示的空间 36 杆桁架，弹性模量 $E=2.1\times10^{11}\text{Pa}$，质量密度 $\rho=7.8\times10^{3}\text{kg/m}^{3}$. 在节点 12 的 $-x$ 方向施加一大小为 50 000N 的力. 约束条件为：结构的基频约束要求基频不小于 80Hz，应力约束要求每个单元的应力不大于 173.2MPa，静位移约束为节点 12 的 x 方向位移小于 0.001m. 通过遗传算法最终优化的拓扑如图 8.6 所示，相应构件的横截面积和优化的特性分别见表 8.2 和表 8.3.

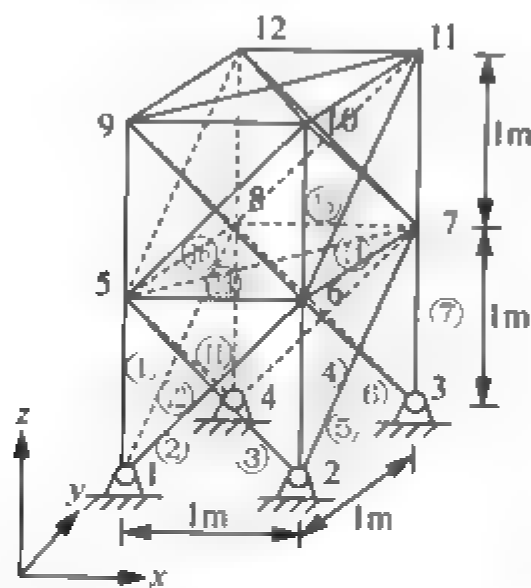


图 8.5　空间 36 杆桁架

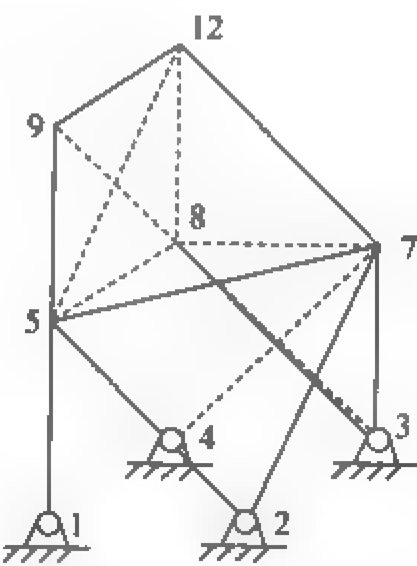


图 8.6　优化的拓扑

表 8.2　优化的横截面积

杆　号	横截面积(mm²)
1	1125
3	672 2
5	180
7	1620
8	320
9	320
11	259 2

续表

杆 号	横截面积(mm^2)
15	80
16	80
17	125
19	3125
26	1620
28	320
29	1620
30	672 2
34	259 2

表 8.3 优化的结构特性

f(Hz)	σ_{max}(MPa)	x_{10max}(mm)	W(kg)
89 454	156.25	0 74405	118 46

```
% ---------------------------------------------------------------------
% Ex 8.3
% This program is used for the topology optimization for space 36-bar trusss
%  based on genetic algorithm
% ---------------------------------------------------------------------
% initialization of parameters
% ---------------------------------------------------------------------
  LENGTH=[4 7];                 %the chromosome length of design variable
  gan=[36 1];                   %the number of gan
  CHROMLENGTH=LENGTH*gan';      %totle length of chromosome
  PopSize=400;                  %population size
  Pc=0.8;                       %probability of crossover
  Pm=0.02;                      %probability of mutation
  penal=100;                    %the initial penalty
%   ------------------------------------------------------
%  the definition of data structure
%   ------------------------------------------------------
 for loopi=1:PopSize
    for loopj=1:CHROMLENGTH
       populationchrom(loopi,loopj)=0.0;
    end
 end
 for loopi=1:3
     for loopj=1:CHROMLENGTH
        bestworstchrom(loopi,loopj)=0.0;
     end
```

```
end
%----    ---    ---    --    --    --    --    --    -
%   begin the main program
%----    --    --    --    --    ---    --    -    -
populationchrom=GenerateInitialPopulation(PopSize,CHROMLENGTH,LENGTH,gan);
populationfitness=CalculateObjectValue(populationchrom,LENGTH,PopSize,
gan,penal);
bestworstchrom=FindBestAndWorstIndividual(populationchrom,
populationfitness,PopSize,bestworstchrom);
bestworstchrom(3,:)=bestworstchrom(1,:);
%-------the ceasing condition ---   --   --
itercri=1;
generation=0;
while itercri

  populationchrom=GenerateNextPopulation(PopSize,populationchrom,
  CHROMLENGTH,LENGTH,Pc,Pm,populationfitness);
populationfitness=CalculateObjectValue(populationchrom,LENGTH,PopSize,
gan,penal);
bestworstchrom=FindBestAndWorstIndividual(populationchrom,
populationfitness,PopSize,bestworstchrom);
  for i=1:3
    [Infor_node,area]=Decode(bestworstchrom(i,:),LENGTH,gan);
    [W,dispmax,stressmax,frequency]=Trusstopology(Infor_node,area);
    C=VALUE(W,dispmax,stressmax,frequency,penal);
    bestworstfitness(i)=C
  end
  if bestworstfitness(1) > bestworstfitness(3)
    bestworstchrom(3,:)=bestworstchrom(1,:);
  end
%----output the result of current population----- ---
  [Infor_node,area]=Decode(bestworstchron(3,:),LENGTH,gan);
  [W,dispmax,stressmax,frequency]=Trusstopology(Infor_node,area);
  if penal>=2
    penal=penal/2;
  end
%----renew the best individual value------------ --
  C=VALUE(W,dispmax,stressmax,frequency,penal);
  bestworstfitness(3)=C;
  sumfitness=0.0;
  for i=1:PopSize
    sumfitness=sumfitness+populationfitness(i);
  end
  meanfitness=(sumfitness-min(populationfitness)+bestworstfitness(3))/PopSize,
  generation=generation+1
  maxfitdata(generation)=bestworstfitness(3);
  meanfitdata(generation)=meanfitness;
  Edata(generation)=W;
```

```
    if generation>50
        h1=abs(Edata(generation)-Edata(generation-50))/Edata(generation);
        if h1<1e-4& frequency>=80&stressmax<=173.2e6&dispmax<=1e-3
            itercri=0;
        else
            itercri=1;
        end
    end
  end
  figure(1)
  plot(Edata)
  figure(2)
  n=1:generation;
  plot(n,maxfitdata,n,meanfitdata,'--')

function [W,dispmax,stressmax,frequency]=Trusstopology(Infor_node,area)
%------------------------------------------------------------------------
% Purpose:
% compute some characteristics corresponding to some truss topology based
% on space 36-bar ground truss
%
% Synosis:
% [W,dispmax,stressmax,frequency]=Trusstopology(Infor_node,area)
%
% Variable Description:
%   Infor_ node - Information on removed nodes
%   area        - cross-section areas
%   W           - total mass
%   dispmax     - maximal static displacement amplitude
%   stressmax   - maximal static stress
%   frequency   - first natural frequency
%------------------------------------------------------------------------
nnoder=find(Infor_node==0)+4;          % Information on removed nodes
E=2.1e11;                  % elastic modulus
density=7860;               % mass density
node_number=12;            % total number of nodes in system before remove
element_number=36;          % number of elements before remove
No_dof=3;                  % number of dofs per node
No_nel=2;                  % number of nodes per element
Sdof=node_number*No_dof;      % total number of dofs for ground structure
f=zeros(Sdof,1);              % force vector
f(34)=-50000;
nc=[0 0 0;1 0 0;1 1 0;0 1 0; ...   % nodal coordinate for ground structure
   0 0 1;1 0 1;1 1 1;0 1 1; ...
   0 0 2;1 0 2;1 1 2;0 1 2];
i=-1;
for loopi=1:length(nnoder)        % obtain new nodal coodinate values
```

```
    i=i+1;
    nc((nnoder(loopi)-i),:)=[];
  end

  en=zeros(36,2);
  en(1,1)=1;en(1,2) 5;   % nodal connectivity for each element beofore remove
  en(2,1)=1;en(2,2)=6;
  en(3,1)=2;en(3,2)=5;
  en(4,1)=2;en(4,2)=6;
  en(5,1)=2;en(5,2)=7;
  en(6,1)=3;en(6,2)=6;
  en(7,1)=3;en(7,2)=7;
  en(8,1)=3;en(8,2)=8;
  en(9,1)=4;en(9,2)=7;
  en(10,1)=4;en(10,2)=8;
  en(11,1)=4;en(11,2)=5;
  en(12,1)=1;en(12,2)=8;
  en(13,1)-5;en(13,2)=6;
  en(14,1)=6;en(14,2)=7;
  en(15,1)-7;en(15,2)=8;
  en(16,1)=5;en(16,2)=8;
  en(17,1)=6;en(17,2)=8;
  en(18,1)=5;en(18,2)=7;
  for loopi=1:1
     for loopj=1:18
        en(loopi*18+loopj,1)=en(loopj,1)+4*loopi;
        en(loopi*18+loopj,2)=en(loopj,2)+4*loopi;
     end
  end
    nelementr=zeros(element_number,1);
     for loopi=1:element_number
         for loopj=1:2
             for loopk=1:length(nnoder)
               if en(loopi,loopj)~=nnoder(loopk)
                    nelementr(loopi)=1;
               end
             end
         end
     end

     num1=0;
     for loopi 1:element_number
        if nelementr(loopi)==1
           num1-num1+1;
           en((loopi-num1+1),:) [];
           area(loopi-num1+1)=[];
        end
```

```
    end

  num2=size(en);
  element_number=num2(1);
  node_number=node_number-length(nnoder);
  flag1=1;
  if element_number<(3*(node_number-4)-1) % judege whether the truss is mechanic
     flag1=0;
  else
  for loopi=1:element_number
     for loopj=1:2
        num3=0;
        for loopk=1:length(nnoder)
           if en(loopi,loopj)>=nnoder(loopk)
              num3=num3+1;
           end
        end
        en(loopi,loopj)=en(loopi,loopj)-num3;
     end
  end
Sdof=Sdof-No_dof*length(nnoder);        % new system dofs
ed(1:node_number,1:3)=1;                % initilizate
element_displacement
constraint=[1,1;1,2;1,3;2,1;2,2;2,3;3,1;3,2;3,3;4,1;4,2;4,3];
for loopi=1:length(constraint);
   ed(constraint(loopi,1),constraint(loopi,2))=0;.
end
dof=0;
for loopi=1:node_number
     for loopj=1:3
        if ed(loopi,loopj)~=0
             dof=dof+1;
             ed(loopi,loopj)=dof;
        end
     end
end
%-----------------------------------------
% Initilization to zero
%-----------------------------------------
k(1:dof,1:dof)=0;                  %structural_stiffness_matrix
m(1:dof,1:dof)=0;;                 %structural_mass_matrix
disp(1:dof)=0;                    % system displacement vector
eldisp(1:No_nel*No_dof)=0;          % element nodal displacement vector
elforce(1:No_nel*No_dof)=0;         % element force vector
stress(1:element_number)=0;          % stress vector for every element
el(1:element_number)=0;
e2s(1:6)=0;  % index of transform the element displament number to structural
```

```
%--------------------------------------------------------------
% Assemble system mass and siffness matrix
%--------------------------------------------------------------
W=0;
for loopi=1:element_number
    for zi=1:2
        e2s((zi-1)*2+1)=ed(en(loopi,zi),1);
        e2s((zi-1)*2+2)=ed(en(loopi,zi),2);
        e2s((zi-1)*2+3)=ed(en(loopi,zi),3);
    end

el(loopi)=sqrt((nc(en(loopi,1),1)-nc(en(loopi,2),1))^2+(nc(en(loopi,1),2)
-nc(en(loopi,2),2))^2 ...
            +(nc(en(loopi,1),3)-nc(en(loopi,2),3))^2);
    c=(nc(en(loopi,1),1)-nc(en(loopi,2),1))/el(loopi);
    s=(nc(en(loopi,1),2)-nc(en(loopi,2),2))/el(loopi);
    r=(nc(en(loopi,1),3)-nc(en(loopi,2),3))/el(loopi);

    dk=E*area(loopi)/el(loopi)*[c^2 c*s c*r -c^2 -c*s -c*r; ...
    % element stiffness matrix
                        c*s s^2 s*r -c*s -s^2 -s*r; ...
                        c*r s*r r^2 -c*r -s*r -r^2; ...
                        -c^2 -c*s -c*r c^2 c*s c*r; ...
                        -c*s -s^2 -s*r c*s s^2 s*r; ...
                        -c*r -s*r -r^2 c*r s*r r^2];
    dm=(density*area(loopi)*el(loopi))/2*eye(6);   % element mass matrix
    for jx=1:6
        for jy=1:6
            if(e2s(jx)*e2s(jy)~=0)
               k(e2s(jx),e2s(jy))=k(e2s(jx),e2s(jy))+dk(jx,jy);
               m(e2s(jx),e2s(jy))=m(e2s(jx),e2s(jy))+dm(jx,jy);
            end
        end
    end
    W=W+density*area(loopi)*el(loopi);
end
if det(k)<1e4        % judge whether system stiffness matrix is singular
    flag1=0;
else
%--------------------------------------------------------------
% Compute the first natural frequency
%--------------------------------------------------------------
[v,d]=eig(k,m); %d:eigenvalue  v:eigenvector
freq=sqrt(diag(d))/(2*pi);
[freq,indexf]=sort(freq);
%--------------------------------------------------------------
% Compute system displacement
```

```
%   -------       --------      --------
i= 1;
for loopi=1:length(nnoder)    % consider nodal remove
   for loopj=1:3
        i=i+1;
        f((nnoder(loopi)-1)*3+loopj-i,:)=[];
   end
end
 Number_con=-1;               % apply boundary conditions
 for loopi=1:length(constraint)
     Number_con= Number_con+1;
     f((constraint(loopi,1)-1)*No_dof+constraint(loopi,2)-Number_con)=[];
 end
 disp=k\f;                   % solve matrix equation for nodal displacement
%----------------------    ---------------
% post computation for stress calculation
%----------  - --------------------------
  for loopi=1:element_number
   for zi=1:2
       e2s((zi-1)*2+1)=ed(en(loopi,zi),1);
       e2s((zi-1)*2+2)=ed(en(loopi,zi),2);
       e2s((zi-1)*2+3)=ed(en(loopi,zi),3);
   end
el(loopi)=sqrt((nc(en(loopi,1),1)-nc(en(loopi,2),1))^2+(nc(en(loopi,1),2)
-nc(en(loopi,2),2))^2 ...
            +(nc(en(loopi,1),3)-nc(en(loopi,2),3))^2);
   c=(nc(en(loopi,1),1)-nc(en(loopi,2),1))/el(loopi);
   s=(nc(en(loopi,1),2)-nc(en(loopi,2),2))/el(loopi);
   r=(nc(en(loopi,1),3)-nc(en(loopi,2),3))/el(loopi);

   dk=E*area(loopi)/el(loopi)*[c^2 c*s c*r -c^2 -c*s -c*r; ...
                              c*s s^2 s*r -c*s -s^2 -s*r; ...
                              c*r s*r r^2 -c*r -s*r -r^2; ...
                              -c^2 -c*s -c*r c^2 c*s c*r; ...
                              -c*s -s^2 -s*r c*s s^2 s*r; ...
                              -c*r -s*r -r^2 c*r s*r r^2];

   for loopj=1:(No_nel*No_dof)
      if e2s(loopj)==0
         eldisp(loopj)=0;
      else
         eldisp(loopj)=disp(e2s(loopj));
      end
   end
    elforce=dk*eldisp';                          % element force vector
   if area(loopi)==0
      stress(loopi)=0;
```

```
      else
         stress(loopi)=sqrt(elforce(1)^2+elforce(2)^2+elforce(3)^2)/area(loopi);
  % stress
      end
      if ((nc(en(loopi,1),1)-nc(en(loopi,2),1))*elforce(4))<0
      % check if tension or compression
         stress(loopi)=-stress(loopi);
      end
    end
  % --- --- --- ---- -------- ---- -------
  % Construct the taotal displacement
  %--- --- ---------------------- ------
  flag2=1;
  iter=-1;
  displ=disp;
  while flag2
      for loopi=1:length(constraint)
          iter=iter+1;
          for loopj=1:(dof+iter)
              if loopj<((constraint(loopi,1)-1)*No_dof+constraint(loopi,2))
                  dispt(loopj)=displ(loopj);
              else
                  dispt(loopj+1)=displ(loopj);
              end
          end
          dispt((constraint(loopi,1)-1)*No_dof+constraint(loopi,2))=0;
          displ=dispt;

      end
      if length(displ)>=node_number*No_dof
            flag2=0;
      end
  end
  %------- ----------------------------------------
  %  Output solutions
  %-----------------------------------------------
   dispmax=max(abs(dispt'));
   stressmax=max(abs(stress'));
   frequency=freq(1);
   end
   end
   if flag1=_0;
      W=1e10;
      dispmax=1;
      frequency=0;
      stressmax-2e10;
  end
 %- -- --- -- -- -- -- --- -- -- --- -- -- --- -- -- -- - -- -- -- -
```

2. 连续体拓扑优化

【例 8.4】 数值算例. 基结构如图 8.7 所示，为 240mm×100mm 的平面体，厚度为 6mm，材料弹性模量为 68.89GPa，泊松比为 0.3，密度为 10 000kg/m³. 一个集中载荷 $F=15\,600$N 作用于右边界中心，为了避免应力集中的影响，将载荷分散在右边界中心的 3 个节点上，左边界固定支承，共划分为 20×48 个四边形网格. 约束：位移约束 P 点竖直向下位移小于 0.15mm，同时结构基频大于 3000Hz. 优化的拓扑以及结构质量随迭代次数的变化曲线分别如图 8.8 和图 8.9 所示.

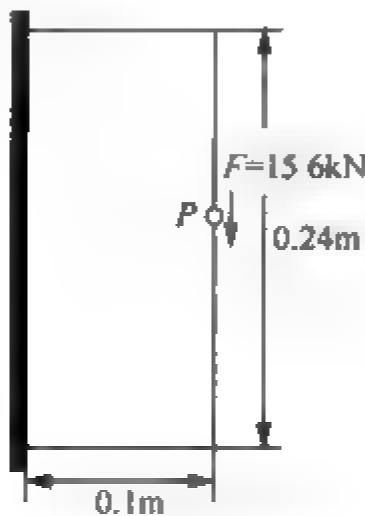


图 8.7　板的基结构

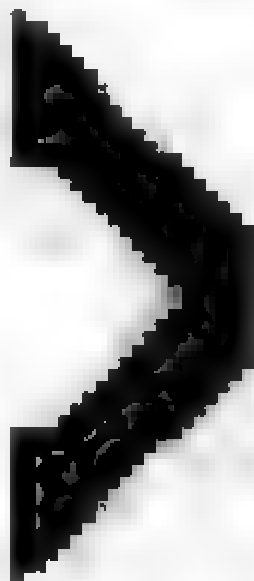


图 8.8　优化的拓扑

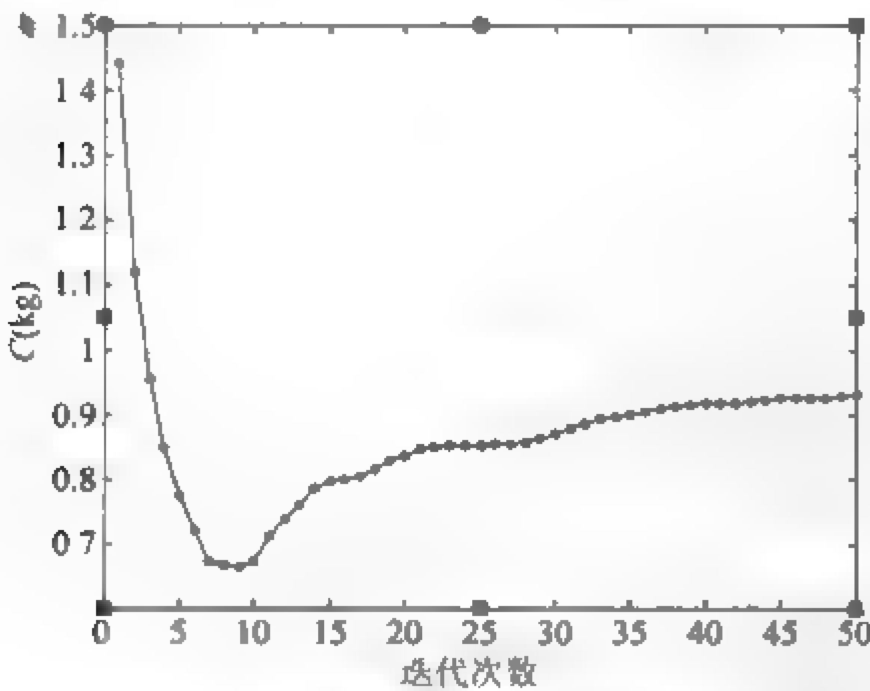


图 8.9　结构质量随迭代次数的变化曲线

```
%----------------------------------------------------------------------
% Ex.8.4
% one side with fixly supported condition for rectangular plate with individual
% displacement and frequency constraints
%----------------------------------------------------------------------
% INITIALIZE
```

```
 nelx 20;  % the numbers of elements of x direction
 nely=48;  % the numbers of  elements of y direction
 x(1:nely,1:nelx) = 0.006*ones(nely,nelx);
 %  initialization of thickness of elements
 a=0.0025;  % length of element along x direction
 b=0.0025;  % length of element along y direction
 p=1e4;     %  density of material
 loop = 0;   % iteration variable
 h=0.2;     % relaxation factor
 U44=-0.00015;  % constrained displacement
 freq1=3000;    % constrained frequency
 change =0.01;
 q=0.21;      % parameter1
 d=1.4;       % parameter2 about frequency sensitivity
 g=0.1;       % parameter3 about frequency sensitivity
 % START ITERATION
 while change>0.0001
  loop = loop + 1;
   xold = x;
 % FE-ANALYSIS
  [U,U2,G,freq,lambda]=FE(nelx,nely,xold);
 % OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS

 [KE,ME] = lk;
    c = 0.;
   for ely = 1:nely
     for elx = 1:nelx
       n1 = (nely+1)*(elx-1)+ely;
       n2 = (nely+1)* elx   +ely;
       edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1; 2*n2+2; 2*n1+1; 2*n1+2];
       c = c + 4*((xold(ely,elx)))*p*a*b;
       dc(ely,elx) = 4*p*a*b;
       Ue = U([2*n1-1; 2*n1; 2*n2 1; 2*n2; 2*n2+1; 2*n2+2; 2*n1+1; 2*n1+2],1);
       F2(edof,:)=xold(ely,elx)*KE*Ue;
       dU(ely,elx)=F2(edof,:)'*U2([edof],2*(nelx+0.5)*(nely+1)+1);
       end
  end
% Lagrange multiplier
   lamda=c/(3*abs(U44));
% COMPUTATION OF DISPLACEMENT SENSITIVITIES
 [U5]=reddistrib(nelx,nely,x,dU,lamda,p,a,b,h);
% COMPUTATION OF FREQUENCY SENSITIVITIES
 [G,G1,G2]=refdistrib(nelx,nely,G,p,a,b,dU);
% DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
[x]=OC(nelx,nely,x,h,U5,q,G,freq,d,g,freq1,p,a,b,dU,G1,G2);
% iterated objective function
y(loop)=c;
```

```
% PRINT RESULTS
change = abs(4*p*a*b*sum(sum(x))-4*p*a*b*sum(sum(xold)))/abs(4*p*a*b*sum(sum(x)));
  disp([' It.: ' sprintf('%4i',loop) ' Obj.: '
  sprintf('%10.4f',  4*p*a*b*sum(sum(x))) ...
       ' Vol.: ' sprintf('%6.3f',4*sum(sum(x))*a*b) ...
        ' freq(1).: ' sprintf('%6.3f',freq(1) )])

% PLOT DENSITIES
figure(1)
  colormap(gray); imagesc(-x); axis equal; axis tight; axis off;pause(1e-6);
  grid on;
figure(2)
plot(y)
end

function [xnew]=OC(nelx,nely,x,h,U5,q,G,freq,d,g,freq1,p,a,b,dU,G1,G2)
%  -----------------------------------------------------------------------
%  OPTIMALITY CRITERIA UPDATE
%------------------------------------------------------------------------
step1=1;
for ely = 1:nely
    for elx = 1:nelx
     xnew1(ely,elx)=real(x(ely,elx)*U5(ely,elx))^(1/(1+2*h));
     %  update design variables based displacement sensitivity

xnew2(ely,elx)=real((((freq1/(freq(1)))^d)*(((G1/G2)*(G(ely,elx)/((1/dU(
ely,elx)))))^g))*x(ely,elx));
%  update design variables based frequency sensitivity
     TT1((elx-1)*nely+ely)=xnew1(ely,elx);
     TT2((elx-1)*nely+ely)=xnew2(ely,elx);
   end
end


for ely = 1:nely
    for elx = 1:nelx
     xnew2(ely,elx)=xnew2(ely,elx)*(median(TT1)/median(TT2));
     %  tension of different update design variables based different
     %  sensitivities
     xnew(ely,elx)=min(max(xnew1(ely,elx),xnew2(ely,elx)),(xnew1(ely,elx)
     +xnew2(ely,elx))/2);  %  tradeoff of update design variables
     T((elx-1)*nely+ely)=xnew(ely,elx);
   end
end

%  deletion of certain percentage of elements using Evolutionary
%  Structural Optimization(ESO)
```

```
Mt=sort(T);
    for i=1:(nely*nelx)
      if((0.00005)<Mt(i)&Mt(i)<(0.009))
       Maa(step1)=Mt(i);
       step1=step1+1;
      end
    end
TT=2*floor(step1*0.005)
for ely = 1:nely
    for elx = 1:nelx
     if((xnew(ely,elx)>0.000009)&(xnew(ely,elx)<0.009)&TT~=0)
        if(xnew(ely,elx)<=Maa(TT))
         xnew(ely,elx)=0.000009;
        end
     end
     TTT((elx-1)*nely+ely)=xnew(ely,elx);
    end
end

% division of design variable based average value
for ely = 1:nely
    for elx = 1:nelx
     w(ely,elx)=q*sign(xnew(ely,elx)-mean(TTT))*xnew(ely,elx);
     xnew(ely,elx)=xnew(ely,elx)+w(ely,elx);
     xnew(ely,elx)=min(max(xnew(ely,elx),0.000009),0.009);
     if((abs(x(ely,elx)-0.000009)<=1e-5)|(abs(x(ely,elx)-0.009)<=1e-5))
     xnew(ely,elx)=x(ely,elx);
     end
    end
end

function [U5]=reddistrib(nelx,nely,x,dU,lamda,p,a,b,h)
%------------------------------------------------------------------------
% displacement sensitivity and sensitivity redistrubution
%------------------------------------------------------------------------
for ely = 1:nely
    for elx = 1:nelx
      U4(ely,elx)=((lamda/(4*p*a*b))*dU(ely,elx)*x(ely,elx))^h;
 end
end
for ely = 1:nely+1
    for elx = 1:nelx+1
        sum1=0;
        MM1=0;
        aa1=[elx ely;elx-1 ely-1;elx-1 ely;elx ely-1];
        for i=1:4
         if(aa1(i,:)>0&aa1(i,1)<=nelx&aa1(i,2)<=nely)
```

```
          sum1=sum1+U4(aa1(i,2),aa1(i,1));
           MM1=MM1+1;
          end
        end
      U6(ely,elx)=sum1/MM1;
    end
end
for ely = 1:nely
    for elx = 1:nelx
     sum11 0;
     MM11=0;
     aa11=[elx ely;elx+1 ely+1;elx ely+1;elx+1 ely];
      for i=1:4
       if(aa11(i,:)>0&aa11(i,1)<=nelx+1&aa11(i,2)<=nely+1)
       sum11=sum11+U6(aa11(i,2),aa11(i,1));
       MM11=MM11+1;
       end
      end
     U5(ely,elx)=sum11/MM11;
    end
end

function [G,G1,G2]=refdistrib(nelx,nely,G,p,a,b,dU)
%-------------------------------------------------------------------------
% frequency sensitivity and sensitivity redistrubution
%-------------------------------------------------------------------------
G1=0;
G2=0;
for ely = 1:nely+1
    for elx = 1:nelx+1
        sum2=0;
        MM2=0;
        aa2=[elx ely;elx-1 ely-1;elx-1 ely;elx ely-1];
        for i=1:4
         if(aa2(i,:)>0&aa2(i,1)<=nelx&aa2(i,2)<=nely)
          sum2=sum2+G(aa2(i,2),aa2(i,1));
           MM2=MM2+2;
         end
        end
      G3(ely,elx)=sum2/MM2;
    end
end
for ely = 1:nely
    for elx = 1:nelx
     sum21=0;
     MM21 0;
     aa21=[elx ely;elx+1 ely+1;elx ely+1;elx+1 ely];
```

```
        for i=1:4
         if(aa21(i,:)>0&aa21(i,1)<=nelx+1&aa21(i,2)<=nely+1)
         sum21=sum21+G3(aa21(i,2),aa21(i,1));
         MM21=MM21+1;
         end
        end
       G(ely,elx)=sum21/MM21;
      end
  end
  for ely = 1:nely
      for elx = 1:nelx
      G1=G1+(1/dU(ely,elx))*G(ely,elx);
      G2=G2+G(ely,elx)^2;
      end
  end

  function [U,U2,G,freq,lambda]=FE(nelx,nely,xold)
  %-----------------------------------------------------------------
  % FE-ANALYSIS
  %-----------------------------------------------------------------
  [KE,ME] = lk;
  K = sparse(2*(nelx+1)*(nely+1), 2*(nelx+1)*(nely+1));
  M = sparse(2*(nelx+1)*(nely+1), 2*(nelx+1)*(nely+1));
  F = sparse(2*(nely+1)*(nelx+1),1);
  F1=sparse(2*(nely+1)*(nelx+1),2*(nely+1)*(nelx+1));
  U = sparse(2*(nely+1)*(nelx+1),1);
  U2 = sparse(2*(nely+1)*(nelx+1),2*(nely+1)*(nelx+1));
  for elx = 1:nelx
    for ely = 1:nely
      n1 = (nely+1)*(elx-1)+ely;
      n2 = (nely+1)* elx   +ely;
      edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1; 2*n2+2; 2*n1+1; 2*n1+2];
      K(edof,edof) = K(edof,edof) + xold(ely,elx)*KE;
      M(edof,edof) = M(edof,edof) + xold(ely,elx)*ME;
    end
  end

  % DEFINE LOADS AND SUPPORTS
  F(2*(nelx+0.5)*(nely+1)+1,1) = -5200;
  F(2*(nelx+0.5)*(nely+1)-1,1)=-5200;
  F(2*(nelx+0.5)*(nely+1)+3,1)=-5200;
  fixeddofs   = [1:2*(nely+1)];
  alldofs     = [1:2*(nely+1)*(nelx+1)];
  freedofs    = setdiff(alldofs,fixeddofs);
  % SOLVING
  U(freedofs,:) = K(freedofs,freedofs) \ F(freedofs,:);
  U(fixeddofs,:)= 0;
```

```
% compute eigenvalues and eigenvectors
[V,D]=eigs(K(freedofs,freedofs), M(freedofs,freedofs),1,'SM');
[lambda,Kt]=sort(diag(D));
Factor=diag(V(:,1)'*M(freedofs,freedofs)*V(:,1));
Vnorm=V(:,1)*inv(sqrt(diag(Factor)));        % normalize eigenvector
freq=sqrt(lambda)/(2*pi);
V1=zeros(2*(nely+1)*(nelx+1),1);
V1(freedofs,:)=Vnorm(:,1);
V1(fixeddofs,:)=0;                          % delete of zero element
% computation of frequency sensitivity
for elx = 1:nelx
  for ely = 1:nely
    n1 = (nely+1)*(elx-1)+ely;
    n2 = (nely+1)* elx   +ely;
    edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1; 2*n2+2; 2*n1+1; 2*n1+2];
    K1=KE;
    M1=ME;
    G(ely,elx)=[V1(edof,1)'*(K1-lambda(1)*M1)*V1(edof,1)];
   end
end
% compute unit virtual load vector
%-----------------------------------------------------------------
for i=1:2*(nely+1)*(nelx+1)
     if(i==2*(nelx+0.5)*(nely+1)+1)
     F1(i,i)=-1;
     U2([freedofs],i)=K(freedofs,freedofs) \ F1([freedofs],i);
     U2([fixeddofs],i)= 0;
     end
end

function [KE,ME]=lk
%-----------------------------------------------------------
% ELEMENT STIFFNESS AND MASS MATRICES
%-----------------------------------------------------------
E = 6.889e10;
nu = 0.3;
a=0.0025;
b=0.0025;
p=1e4;
k=[ 1/2-nu/6   1/8+nu/8 -1/4-nu/12 -1/8+3*nu/8 ...
   -1/4+nu/12 -1/8-nu/8  nu/6       1/8-3*nu/8];
% stiffness matrix
KE = E/(1-nu^2)*[ k(1) k(2) k(3) k(4) k(5) k(6) k(7) k(8)
                  k(2) k(1) k(8) k(7) k(6) k(5) k(4) k(3)
                  k(3) k(8) k(1) k(6) k(7) k(4) k(5) k(2)
                  k(4) k(7) k(6) k(1) k(8) k(3) k(2) k(5)
                  k(5) k(6) k(7) k(8) k(1) k(2) k(3) k(4)
```

```
              k(6) k(5) k(4) k(3) k(2) k(1) k(8) k(7)
              k(7) k(4) k(5) k(2) k(3) k(8) k(1) k(6)
              k(8) k(3) k(2) k(5) k(4) k(7) k(6) k(1)];
% mass matrix
ME=((p*a*b)/9)*[4 0 2 0 1 0 2 0
               0 4 0 2 0 1 0 2
               2 0 4 0 2 0 1 0
               0 2 0 4 0 2 0 1
               1 0 2 0 4 0 2 0
               0 1 0 2 0 4 0 2
               2 0 1 0 2 0 4 0
               0 2 0 1 0 2 0 4];
%-------------------------------------------------------------------
```

【例 8.5】 基本结构如图 8.10 所示，为520mm×260mm的平面体，厚度为5mm，材料弹性模量为68.89GPa，泊松比为 0.3，密度为1kg/cm^3。一个集中载荷 $F=21\,000$N 作用于下边界中点，为了避免应力集中的影响，将载荷分散在下边界中心的 3 个节点上，左下角和右下角两个点采用固定支承，共划分为52×26个四边形网格. 约束：位移约束 P 点竖直向下位移小于0.5mm，同时结构基频大于70Hz．优化的拓扑以及结构质量随迭代次数的变化曲线分别如图 8.11 和图 8.12 所示.

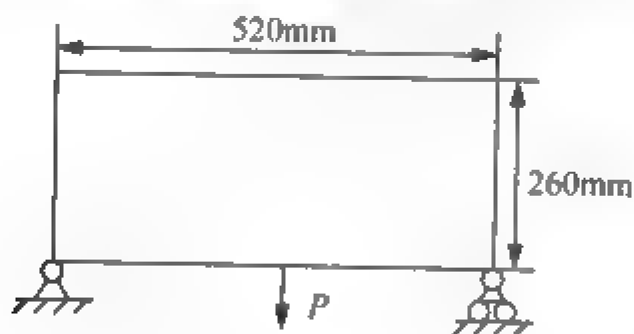


图 8.10 板的基结构

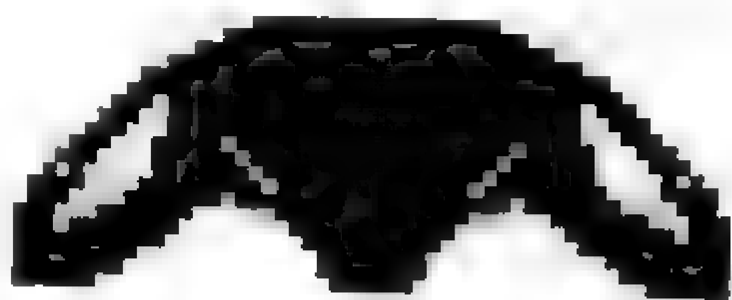


图 8.11 优化的拓扑

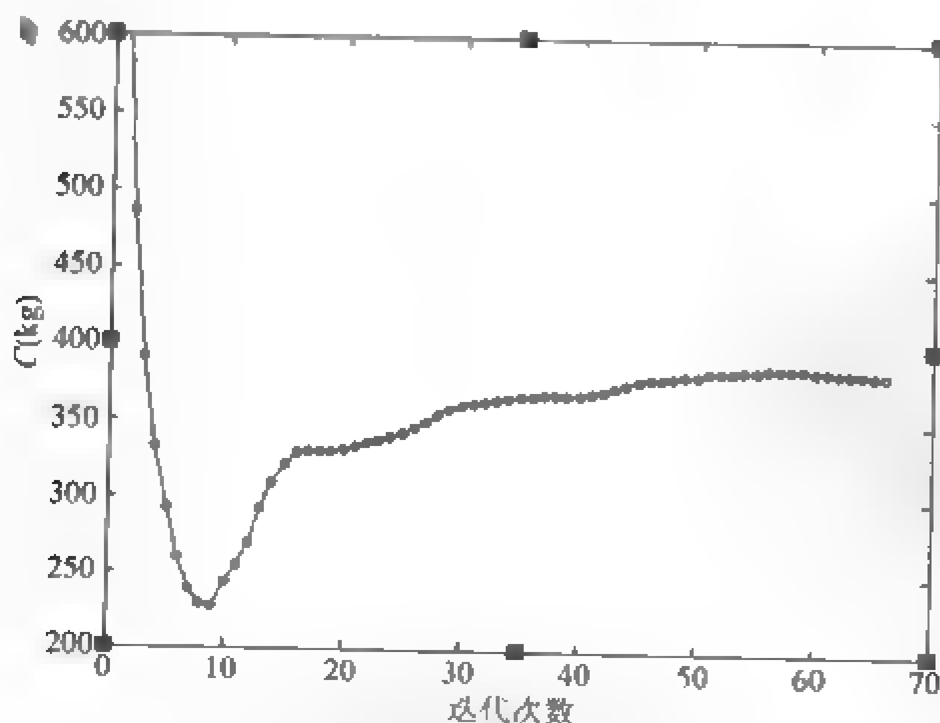


图 8.12 结构质量随迭代次数的变化曲线

```
%-------------------------------------------------------
% Ex.8.5
% one side with fixly-supported condition for rectangular plate with
% individual displacement and frequence constraint
% 30*16 Mesh
%---------------------------------------------------------------
% INITIALIZE
nelx=52;
nely=26;
x(1:nely,1:nelx) = 0.005*ones(nely,nelx);
a=0.005;
b=0.005;
p=1e6;
loop = 0;
h=0.21;
U44=-0.0005;
freq1=70;
change =0.01;
% START ITERATION
q=0.2;
d=1.4;      % parameter 2
g=0.28;      % parameter 3
while change>0.00001
 loop = loop + 1;
  xold = x;
% FE-ANALYSIS

  [U,U2,G,freq,lambda]=FE(nelx,nely,xold);
  if ((loop)>120),break,end
% OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS

[KE,ME] = lk;
    c = 0.;
    U99=U(2*(0.5*nelx+1)*(nely+1),1)
    freq(1)
  for ely = 1:nely
    for elx = 1:nelx
      n1 = (nely+1)*(elx-1)+ely;
      n2 = (nely+1)* elx   +ely;
      edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1; 2*n2+2; 2*n1+1; 2*n1+2];
      c = c + 4*((xold(ely,elx)))*p*a*b;
      dc(ely,elx) = 4*p*a*b;
      Ue = U([2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1; 2*n2+2; 2*n1+1; 2*n1+2],1);
      F2(edof,:)=xold(ely,elx)*KE*Ue;
      dU(ely,elx)=F2(edof,:)'*U2([edof],2*(0.5*nelx+1)*(nely+1));
      end
```

```
    end
  % Lagrange multiplier
     lamda=c/(3*abs(U44));
  % COMPUTATION OF DISPLACEMENT SENSITIVITIES
   [U5]=reddistrib(nelx,nely,x,dU,lamda,p,a,b,h);
   [G,G1,G2]=refdistrib(nelx,nely,G,p,a,b,dU);
  % DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
  [x]=OC(nelx,nely,x,h,U5,q,G,freq,d,g,freq1,p,a,b,dU,G1,G2);

  % iterated objective function
  y(loop)=c;

  % PRINT RESULTS
  change = abs(4*p*a*b*sum(sum(x))-4*p*a*b*sum(sum(xold)))/abs(4*p*a*b*sum(sum(x)));
    disp([' It.: ' sprintf('%4i',loop) ' Obj.: '
  sprintf('%6.4f',4*p*a*b*sum(sum(x))) ...
         ' Vol.: ' sprintf('%6.3f',4*sum(sum(x))*a*b) ...
          ' lamda.: ' sprintf('%6.3f',lamda )])

  % PLOT DENSITIES

  figure(1)
    colormap(gray); imagesc(-x); axis equal; axis tight; axis off;pause(1e-6);
  figure(2)
  plot(y)
  end


function [xnew]=OC(nelx,nely,x,h,U5,q,G,freq,d,g,freq1,p,a,b,dU,G1,G2)
%---------------------------------------------------------------------------
% OPTIMALITY CRITERIA UPDATE
%---------------------------------------------------------------------------
step1=1;
for ely = 1:nely
   for elx = 1:nelx
     xnew1(ely,elx)=real(x(ely,elx)*U5(ely,elx))^(1/(1+2*h));
xnew2(ely,elx)=real((((freq1/(freq(1)))^d)*(((G1/G2)*(G(ely,elx)/((1/dU(
ely,elx)))))^g))*x(ely,elx));
     TT1((elx-1)*nely+ely)=xnew1(ely,elx);
     TT2((elx-1)*nely+ely)=xnew2(ely,elx);
   end
end


for ely = 1:nely
    for elx = 1:nelx
     xnew2(ely,elx)=xnew2(ely,elx)*(median(TT1)/median(TT2));
```

```
xnew(ely,elx)=min(max(xnew1(ely,elx),xnew2(ely,elx)),(xnew1(ely,elx)+xne
w2(ely,elx))/2);
     T((elx-1)*nely+ely)=xnew(ely,elx);
   end
end
 Mt-sort(T);
     for i=1:(nely*nelx)
       if((0.00006)<Mt(i)&Mt(i)<(0.006))
        Maa(step1)=Mt(i);
        step1=step1+1;
       end
     end
TT=2*floor(step1*0.01)
for ely = 1:nely
    for elx = 1:nelx
     if((xnew(ely,elx)>0.00006)&(xnew(ely,elx)<0.006)&TT~=0)
        if(xnew(ely,elx)<=Maa(TT))
         xnew(ely,elx)=0.00006;
        end
     end
     TTT((elx-1)*nely+ely)=xnew(ely,elx);
    end
 end
for ely = 1:nely
    for elx = 1:nelx
     w(ely,elx)=q*sign(xnew(ely,elx)-mean(TTT))*xnew(ely,elx);
     xnew(ely,elx)=xnew(ely,elx)+w(ely,elx);
     xnew(ely,elx)=min(max(xnew(ely,elx),0.00001),0.006);
     if((abs(x(ely,elx)-0.00001)<=1e-5)|(abs(x(ely,elx)-0.006)<=1e-5))
     xnew(ely,elx)=x(ely,elx);
     end
    end
 end


function [U,U2,G,freq,lambda]=FE(nelx,nely,xold)
%-----------------------------------------------------------------------
% FE-ANALYSI
%-----------------------------------------------------------------------
[KE,ME]=lik;
K=sparse(2*(nelx+1)*(nely+1), 2*(nelx+1)*(nely+1)); M =
sparse(2*(nelx+1)*(nely+1), 2*(nelx+1)*(nely+1));
F = sparse(2*(nely+1)*(nelx+1),1);
F1=sparse(2*(nely+1)*(nelx+1),2*(nely+1)*(nelx+1));
U = sparse(2*(nely+1)*(nelx+1),1);
U2 =sparse(2*(nely+1)*(nelx+1),2*(nely+1)*(nelx+1));
```

```
F4-sparse(2*(nely+1)*(nelx+1),1);  F5-sparse(2*(nely+1)*(nelx+1),1);
for elx = 1:nelx
  for ely = 1:nely
    n1 = (nely+1)*(elx-1)+ely;
    n2 = (nely+1)* elx   +ely;
    edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1; 2*n2+2; 2*n1+1; 2*n1+2];
    K(edof,edof) = K(edof,edof) + xold(ely,elx)*KE;
    M(edof,edof) = M(edof,edof) + xold(ely,elx)*ME;
  end
end

% DEFINE LOADS AND SUPPORTS (HALF MBB-BEAM)

F(2*(0.5*nelx+1)*(nely+1),1) = -7000;
F(2*(0.5*nelx)*(nely+1),1) = -7000;
F(2*(0.5*nelx+2)*(nely+1),1) = -7000;
fixeddofs   = union([2*(nely+1)-1 2*(nely+1)],[2*(nelx+1)*(nely+1)-1
2*(nelx+1)*(nely+1)]);
alldofs     = [1:2*(nely+1)*(nelx+1)];
freedofs    = setdiff(alldofs,fixeddofs);
% SOLVING

U(freedofs,:) = K(freedofs,freedofs) \ F(freedofs,:);
U(fixeddofs,:)= 0;
% compute eigenvalues and eigenvectors
[V,D]=eigs(K(freedofs,freedofs), M(freedofs,freedofs),1,'SM');
[lambda,Kt]=sort(diag(D));
Factor=diag(V(:,1)'*M(freedofs,freedofs)*V(:,1));
Vnorm=V(:,1)*inv(sqrt(diag(Factor)));          % normalize eigenvector
freq=sqrt(lambda)/(2*pi);
V1=zeros(2*(nely+1)*(nelx+1),1);
V1(freedofs,:)=Vnorm(:,1);
V1(fixeddofs,:)=0;
% computation of frequence sensitivity
for elx = 1:nelx
  for ely = 1:nely
    n1 = (nely+1)*(elx-1)+ely;
    n2 = (nely+1)* elx   +ely;
    edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1; 2*n2+2; 2*n1+1; 2*n1+2];
    K1=KE;
    M1=ME;
    G(ely,elx)=[V1(edof,1)'*(K1-lambda(1)*M1)*V1(edof,1)];
  end
end

% compute eigenvalues and eigenvectors
% --- --- -- --- --- -- --- -- -- -- --- --- --- -- --
```

```
for i=1:2*(nely+1)*(nelx+1)
     if(i==2*(0.5*nelx+1)*(nely+1))
     F1(i,i)=-1;
     U2([freedofs],i)=K(freedofs,freedofs) \ F1([freedofs],i);
     U2([fixeddofs],i)= 0;
    end
   % U1(:,[fixeddofs])=0;
end

% for elx=1:nelx
 %  for ely=1:nely
    % n1=(nely+1)*(elx-1)+ely;
   % n2=(nely+1)* elx+ely;
   % edof=[3*n1-2;3*n1-1;3*n1;3*n2-2;3*n2-1;3*n2;3*n2+1;3*n2+2;3*n2+3;
   % 3*n1+1;3*n1+2;3*n1+3];
   % for i=1:3*(nely+1)*(nelx+1)
   %  if(i==3*(0.5*nelx*(nely+1)+0.5*nely)+1)
    %   U2([edof],i)=U1([3*n1-2;*n1-1;3*n1;3*n2-2;3*n2-1;3*n2;3*n2+1;
    %   3*n2+2;3*n2+3;3*n1+1;3*n1+2;3*n1+3],i);
     %   end
     %  end
  %  end
 % end
function [KE,ME]=llk
%-----------------------------------------------------------------
% ELEMENT STIFFNESS  AND MASS MATRICES
%-----------------------------------------------------------------
E = 68.89e9;
nu = 0.3;
a=0.005;
b=0.005;
p=1e6;
k=[ 1/2-nu/6   1/8+nu/8 -1/4-nu/12 -1/8+3*nu/8 ...
   -1/4+nu/12 -1/8-nu/8  nu/6      1/8-3*nu/8];
KE = E/(1-nu^2)*[ k(1) k(2) k(3) k(4) k(5) k(6) k(7) k(8)
                  k(2) k(1) k(8) k(7) k(6) k(5) k(4) k(3)
                  k(3) k(8) k(1) k(6) k(7) k(4) k(5) k(2)
                  k(4) k(7) k(6) k(1) k(8) k(3) k(2) k(5)
                  k(5) k(6) k(7) k(8) k(1) k(2) k(3) k(4)
                  k(6) k(5) k(4) k(3) k(2) k(1) k(8) k(7)
                  k(7) k(4) k(5) k(2) k(3) k(8) k(1) k(6)
                  k(8) k(3) k(2) k(5) k(4) k(7) k(6) k(1)];
ME=((p*a*b)/9)*[4 0 2 0 1 0 2 0
                0 4 0 2 0 1 0 2
                2 0 4 0 2 0 1 0
                0 2 0 4 0 2 0 1
                1 0 2 0 4 0 2 0
```

```
        0 1 0 2 0 4   2
        2 0 1 0 2 0 4 0
        0 2 0 1 0 2 0 4];
% ------------------------------------------------------------------
```

【例 8.6】 尺寸为6000mm×3600mm的板结构，厚度为500mm，材料弹性模量为210GPa，泊松比为 0.3，密度为7800kg/m³. 一个集中载荷$F=40\,000$N作用于结构中心点，板四周简支，共划分为30×18个四边形网格. 约束：位移约束P点竖直向下位移小于2.8mm，同时结构基频大于3.5Hz. 优化的拓扑以及结构质量随迭代次数的变化曲线分别如图8.13和图8.14所示.

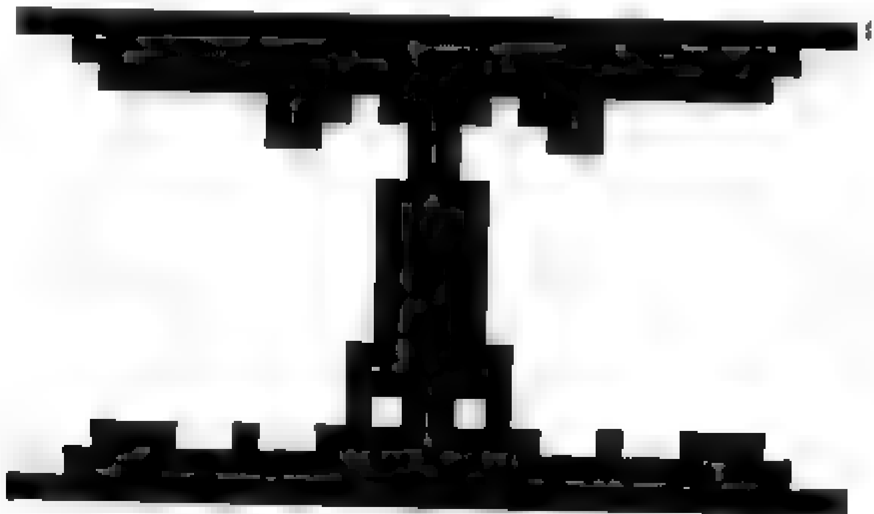


图8.13 优化的拓扑

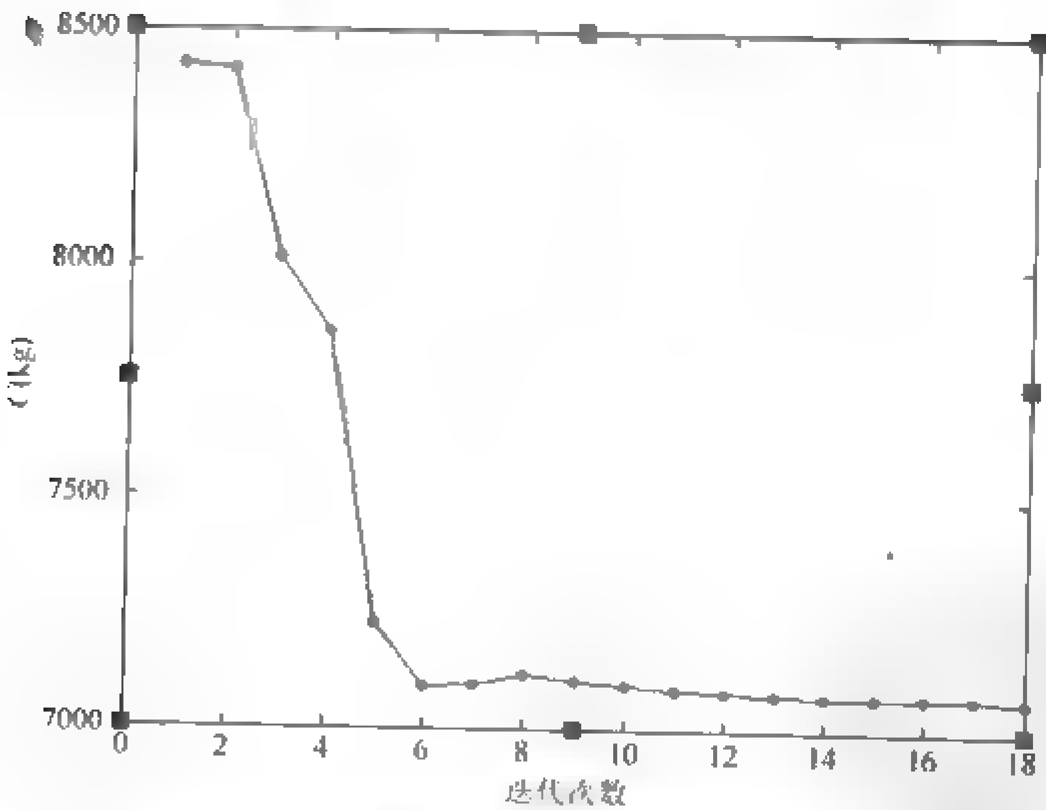


图8.14 结构质量随迭代次数的变化曲线

```
% ------------------------------------------------------------------
% Ex. 8.6
% four  angular point with hinged supported condition for square plate
% with multidisplacement constraints
```

```
%---------------------------------------- ----------------------------
% INITIALIZE
nelx=30;
nely_18;
x(1:nely,1:nelx) = 0.05*ones(nely,nelx);
a=0.1;
b-0.1;
p 7800;
loop = 0;
h=0.5;
U4=-0.01;
change =0.02;
q=0.4;
U6=-0.0028;
d=1.4;
g=0.31;
freq1=3.5;
% START ITERATION
while change>0.0001
 loop = loop + 1;
  xold = x;
% FE-ANALYSIS
  [U,U2,G,freq,lambda]=FE(nelx,nely,xold);
  if ((loop)>28),break,end
% OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
   [KE,Me] = lk1;
    c = 0.;
    U(3*(0.5*nelx*(nely+1)+0.5*nely)+1,1)
    U(3*(0.5*nelx*(nely+1)+0.5*nely)-2,1)
    freq(1)
  for ely = 1:nely
    for elx = 1:nelx
      n1 = (nely+1)*(elx-1)+ely;
      n2 = (nely+1)* elx   +ely;
      edof = [3*n1-2;3*n1-1;3*n1;3*n2-2;3*n2-1;
      3*n2;3*n2+1;3*n2+2;3*n2+3;3*n1+1;3*n1+2;3*n1+3];
      c = c + 4*((xold(ely,elx)))*p*a*b;
      dc(ely,elx) = 4*p*a*b;
      Ue = U([3*n1-2;3*n1-1;3*n1;3*n2-2;3*n2-1;
      3*n2;3*n2+1;3*n2+2;3*n2+3;3*n1+1;3*n1+2;3*n1+3],1);
      F2(edof,:)=((xold(ely,elx)^3)/((0.01)^3))*KE*Ue;
      dU(ely,elx)=F2(edof,:)'*U2([edof],3*(0.5*nelx*(nely+1)+0.5*nely)+1);
      dU1(ely,elx)=F2(edof,:)'*U2([edof],3*(0.5*nelx*(nely+1)+0.5*nely)-2);
    end
 end
  lamda1(loop)=c/(3*abs(U4));
  lamda2(loop)=c/(3*abs(U6));
```

```
    lamda1(loop+1)=lamda1(loop)*(U(3*(0.5*nelx*(nely+1)+0.5*nely)+1,1)/U4)^h;
    lamda2(loop+1)=lamda2(loop)*(U(3*(0.5*nelx*(nely+1)+0.5*nely)-2,1)/U6)^h;
    lamda5=lamda1(loop);
    lamda6=lamda2(loop);

  % COMPUTATION OF DISPLACEMENT SENSITIVITIES
   [U5]=reddistrib(nelx,nely,x,dU,dU1,lamda5,lamda6,p,a,b,h);
   % COMPUTATION OF FREQUENCE SENSITIVITIES
   [G,G1,G2]=refdistrib(nelx,nely,G,p,a,b,dU,dU1);
  % DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
   [x]=OC(nelx,nely,x,h,U5,q,G,freq,d,g,freq1,p,a,b,dU,dU1,G1,G2);
   % m=1-(1-0.05)^((loop+1)/loop);
  %  m=0.2+0.5^((loop+1)/loop);
  y(loop)=c;
  % PRINT RESULTS
  change = abs(4*p*a*b*sum(sum(x))-4*p*a*b*sum(sum(xold)))/abs(4*p*a*b*sum(sum(x)));
    disp([' It.: ' sprintf('%4i',loop) ' Obj.: '
    sprintf('%10.4f',4*p*a*b*sum(sum(x))) ...
         ' Vol.: ' sprintf('%6.3f',4*sum(sum(x))*a*b) ...
          ' lamda5.: ' sprintf('%6.3f',lamda5 )])
  % PLOT DENSITIES
  figure(1)
    colormap(gray); imagesc(-x); axis equal; axis tight; axis
  off;pause(1e-6);
  figure(2)
  plot(y)
  end

  function
  [xnew]=OC(nelx,nely,x,h,U5,q,G,freq,d,g,freq1,p,a,b,dU,dU1,G1,G2)
  %-------------------------------------------------------------------------
  % OPTIMALITY CRITERIA UPDATE
  %-------------------------------------------------------------------------
  step1=1;
  for ely = 1:nely
      for elx = 1:nelx
       xnew1(ely,elx)=(x(ely,elx)*U5(ely,elx))^(1/(1+2*h));
       xnew2(ely,elx)=real((((freq1/(freq(1)))^d)*(((G1/G2)*(G(ely,elx)/
       ((2/(dU(ely,elx)+dU1(ely,elx))))))^g))*x(ely,elx));
       TT1((elx-1)*nely+ely)=xnew1(ely,elx);
       TT2((elx-1)*nely+ely)=xnew2(ely,elx);
    end
 end

 for ely = 1:nely
     for elx = 1:nelx
      xnew2(ely,elx)=xnew2(ely,elx)*(median(TT1)/median(TT2));
```

```
xnew(ely,elx)=min(max(xnew1(ely,elx),xnew2(ely,elx)),(xnew1(ely,elx)
+xnew2(ely,elx))/2);
T((elx-1)*nely+ely)=xnew(ely,elx);
end
end
Mt=sort(T);
for i=1:(nely*nelx)
if((0.005)<Mt(i)&Mt(i)<(0.1))
Maa(step1)=Mt(i);
step1=step1+1;
end
end
TT=2*floor(step1*0.01)
for ely = 1:nely
for elx = 1:nelx
if((xnew(ely,elx)>0.005)&(xnew(ely,elx)<0.1)&TT~=0)
if(xnew(ely,elx)<=Maa(TT))
xnew(ely,elx)=0.005;
end
end
TTT((elx-1)*nely+ely)=xnew(ely,elx);
end
end

for ely = 1:nely
for elx = 1:nelx
w(ely,elx)=q*sign(xnew(ely,elx)-mean(TTT))*xnew(ely,elx);
xnew(ely,elx)=xnew(ely,elx)+w(ely,elx);
xnew(ely,elx)=min(max(xnew(ely,elx),0.005),0.1);
if((abs(x(ely,elx)-0.005)<=1e-4)|(abs(x(ely,elx)-0.1)<=1e-4))
xnew(ely,elx)=x(ely,elx);
end
end
end

function [U5]=reddistrib(nelx,nely,x,dU,dU1,lamda5,lamda6,p,a,b,h)
%-----------------------------------------------------------------------
% MESH-INDEPENDENCY FILTER
%-----------------------------------------------------------------------
for ely = 1:nely
for elx = 1:nelx
U4(ely,elx)=((lamda5/(4*p*a*b))*dU(ely,elx)*x(ely,elx)+(lamda6/
(4*p*a*b))*dU1(ely,elx)*x(ely,elx))^h;
end
end
for ely = 1:nely+1
for elx = 1:nelx+1
```

```
            sum1=0;
            MM1=0;
            aa1=[elx ely;elx-1 ely-1;elx-1 ely;elx ely-1];
            for i=1:4
             if(aa1(i,:)>0&aa1(i,1)<=nelx&aa1(i,2)<=nely)
              sum1=sum1+U4(aa1(i,2),aa1(i,1));
               MM1=MM1+1;
             end
            end
          U6(ely,elx)=sum1/MM1;
        end
    end
    for ely = 1:nely
        for elx = 1:nelx
         sum11=0;
         MM11=0;
         aa11=[elx ely;elx+1 ely+1;elx ely+1;elx+1 ely];
          for i=1:4
           if(aa11(i,:)>0&aa11(i,1)<=nelx+1&aa11(i,2)<=nely+1)
           sum11=sum11+U6(aa11(i,2),aa11(i,1));
           MM11=MM11+1;
           end
          end
        U5(ely,elx)=sum11/MM11;
       end
   end

   function [G,G1,G2]=refdistrib(nelx,nely,G,p,a,b,dU,dU1)
   %------------------------------------------------------------------------
   % freque sensitivity and sensitivity redistribution
   %------------------------------------------------------------------------
   G1=0;
   G2=0;
   for ely = 1:nely+1
       for elx = 1:nelx+1
           sum2=0;
           MM2=0;
           aa2=[elx ely;elx-1 ely-1;elx-1 ely;elx ely 1];
           for i=1:4
            if(aa2(i,:)>0&aa2(i,1)<=nelx&aa2(i,2)<=nely)
             sum2=sum2+G(aa2(i,2),aa2(i,1));
              MM2=MM2+2;
            end
           end
         G3(ely,elx)=sum2/MM2;
     end
 end
```

```
for ely = 1:nely
    for elx = 1:nelx
     sum21=0;
     MM21-0;
     aa21=[elx ely;elx+1 ely+1;elx ely+1;elx+1 ely];
      for i=1:4
        if(aa21(i,:)>0&aa21(i,1)<-nelx+1&aa21(i,2)<-nely+1)
        sum21=sum21+G3(aa21(i,2),aa21(i,1));
        MM21=MM21+1;
        end
      end
     G(ely,elx)=sum21/MM21;
    end
end
for ely = 1:nely
    for elx = 1:nelx
    G1=G1+(2/(dU(ely,elx)+dU1(ely,elx)))*G(ely,elx);
    G2=G2+G(ely,elx)^2;
    end
end


function [U,U2,G,freq,lambda]=FE(nelx,nely,xold)
%------------ ---------------------------------------------------
% FE-ANALYSIS
%----------------------------------------------------------------
[KE,Me] = lk1;
K = sparse(3*(nelx+1)*(nely+1), 3*(nelx+1)*(nely+1)); M=
sparse(3*(nelx+1)*(nely+1), 3*(nelx+1)*(nely+1));
F = sparse(3*(nely+1)*(nelx+1),1);
F1=sparse(3*(nely+1)*(nelx+1),3*(nely+1)*(nelx+1));
U = sparse(3*(nely+1)*(nelx+1),1);  U1 =
sparse(3*(nely+1)*(nelx+1),3*(nely+1)*(nelx+1));
for elx = 1:nelx
  for ely = 1:nely
    n1 = (nely+1)*(elx-1)+ely;
    n2 = (nely+1)* elx   +ely;
    edof = [3*n1-2;3*n1-1;3*n1;3*n2-2;3*n2-1;3*n2;3*n2+1;3*n2+2;3*n2+3;
    3*n1+1;3*n1+2;3*n1+3];
    K(edof,edof) = K(edof,edof) + ((xold(ely,elx)^3)/((0.01)^3))*KE;
    M(edof,edof) =M(edof,edof)+((xold(ely,elx))/(0.01))*Me;
  end
end
% DEFINE LOADS AND SUPPORTS (HALF MBB-BEAM)
F(3*(0.5*nelx*(nely+1)+0.5*nely)+1,1) = -10000;
fixeddofs  = union([1,3*(nely+1)-2],[3*(nelx)*(nely+1)+1,3*(nelx+1)*(nely+1)-2]);
alldofs     = [1:3*(nely+1)*(nelx+1)];
```

```
freedofs    = setdiff(alldofs,fixeddofs);
% SOLVING
U(freedofs,:) = K(freedofs,freedofs) \ F(freedofs,:);
U(fixeddofs,:)= 0;

% compute eigenvalues and eigenvectors
[V,D]=eigs(K(freedofs,freedofs), M(freedofs,freedofs),1,'SM');
[lambda,Kt]=sort(diag(D));
Factor=diag(V(:,1)'*M(freedofs,freedofs)*V(:,1));
Vnorm V(:,1)*inv(sqrt(diag(Factor)));          % normalize eigenvector
freq=sqrt(lambda)/(2*pi);
V1=zeros(3*(nely+1)*(nelx+1),1);
V1(freedofs,:)-Vnorm(:,1);
V1(fixeddofs,:)-0;
for elx = 1:nelx
  for ely = 1:nely
    n1 = (nely+1)*(elx-1)+ely;
    n2 = (nely+1)* elx   +ely;
    edof - [3*n1-2;3*n1-1;3*n1;3*n2 2;3*n2-1;3*n2;3*n2+1;3*n2+2;3*n2+3;
    3*n1+1;3*n1+2;3*n1+3];
    K1=3*((xold(ely,elx)^2)/((0.01)^3))*KE;
    M1=(1/(0.01))*Me;
    G(ely,elx)=[V1(edof,1)'*(K1-lambda(1)*M1)*V1(edof,1)];
   end
end
%-----------------------------------------------------------------------
for i-1:3*(nely+1)*(nelx+1)
     if((i=-3*(0.5*nelx*(nely+1)+0.5*nely)-2)|(i==3*(0.5*nelx*(nely+1)
     +0.5*nely)+1))
     F1(i,i)=-1;
     U2([freedofs],i)=K(freedofs,freedofs) \ F1([freedofs],i);
% obtain unit virtual load vectors
    U2([fixeddofs],i)= 0;
    end
  %  U1(:,[fixeddofs])=0;
end


function [KE,Me] = lk1
%-----------------------------------------------------------------------
% ELEMENT STIFFNESS AND MASS MATRICES
%-----------------------------------------------------------------------
E = 2.1e11;
miu = 0.3;
a 0.1;
b-0.1;
t 0.01;
```

```
density=7800;
k1=21-6*miu+30*b^2/a^2+30*a^2/b^2;
k2=8*b^2-8*miu*b^2+40*a^2;
k3=8*a^2-8*miu*a^2+40*b^2;
k4=3*b+12*miu*b+30*a^2/b;
k5=3*a+12*miu*a+30*b^2/a;
k6=30*miu*a*b;
k7=-21+6*miu-30*b^2/a^2+15*a^2/b^2;
k8=-8*b^2+8*miu*b^2+20*a^2;
k9=-2*a^2+2*miu*a^2+20*b^2;
k10=-3*b-12*miu*b+15*a^2/b;
k11=3*a-3*miu*a+30*b^2/a;
k12=21-6*miu-15*b^2/a^2-15*a^2/b^2;
k13=2*b^2-2*miu*b^2+10*a^2;
k14=2*a^2-2*miu*a^2+10*b^2;
k15=-3*b+3*miu*b+15*a^2/b;
k16=-3*a+3*miu*a+15*b^2/a;
k17=-21+6*miu+15*b^2/a^2-30*a^2/b^2;
k18=-2*b^2+2*miu*b^2+20*a^2;
k19=-8*a^2+8*miu*a^2+20*b^2;
k20=3*b-3*miu*b+30*a^2/b;
k21=-3*a-12*miu*a+15*b^2/a;
KE = E*t^3/(360*(1-miu^2)*a*b)*...
   [k1  k4  -k5  k7  k10  -k11  k12  k15  -k16  k17  k20  -k21;
   k4   k2  -k6  k10 k8   0   -k15  k13   0   -k20  k18   0;
   -k5  -k6  k3  k11 0    k9   k16  0     k14  -k21  0    k19;
   k7   k10  k11 k1  k4   k5   k17  k20   k21  k12   k15  k16;
   k10  k8   0   k4  k2   k6   -k20 k18   0    -k15  k13  0;
   -k11  0   k9  k5  k6   k3   k21  0     k19  -k16  0    k14;
   k12 -k15  k16 k17 -k20 k21   k1  -k4   k5    k7   -k10 k11;
   k15  k13   0  k20 k18  0    -k4   k2   -k6   -k10  k8   0;
   -k16  0   k14 k21  0   k19   k5  -k6   k3    -k11  0    k9;
   k17 -k20 -k21 k12 -k15 -k16  k7  -k10  -k11   k1   -k4  -k5;
   k20  k18   0  k15  k13  0   -k10  k8    0    -k4   k2   k6;
   -k21 0    k19 k16   0  k14   k11  0     k9   -k5   k6   k3];

Me=zeros(12);
mass=density*(2*a)*(2*b)*t/25200;
Me(1,1)=mass*3454;    Me(2,1)=mass*(922*b);  Me(3,1)=mass*(-922*a);
Me(4,1)=mass*1226;     Me(5,1)=mass*(398*b);    Me(6,1)=mass*(548*a);
Me(7,1)=mass*394;      Me(8,1)=mass*(-232*b);  Me(9,1)=mass*(232*a);
Me(10,1)=mass*1226;    Me(11,1)=mass*(-548*b);  Me(12,1)=mass*(-398*a);

Me(2,2)=mass*(320*b^2);Me(3,2)=mass*(-252*a*b);Me(4,2)=mass*(398*b);
Me(5,2)=mass*(160*b^2); Me(6,2)=mass*(168*a*b);
Me(7,2)=mass*(232*b);  Me(8,2)=mass*(-120*b^2);Me(9,2)=mass*(112*a*b);Me(10,2)
=mass*(548*b); Me(11,2)=mass*(-240*b^2);Me(12,2)=mass*(-168*a*b);
```

```
Me(3,3)=mass*(320*a^2);Me(4,3)=-Me(6,1);      Me(5,3)=Me(12,2);
Me(6,3)=mass*(-240*a^2);Me(7,3)=-Me(9,1);      Me(8,3)=Me(9,2);
Me(9,3)=mass*(-120*a^2);Me(10,3)=Me(12,1);     Me(11,3)=Me(6,2);
Me(12,3)=Me(3,3)/2;

Me(4,4)=Me(1,1);          Me(5,4)=Me(2,1);          Me(6,4)=-Me(3,1);
Me(7,4)=mass*1226;           Me(8,4)=Me(11,1);            Me(9,4)=-Me(12,1);
Me(10,4)=Me(7,1);          Me(11,4)=Me(8,1);          Me(12,4)=-Me(9,1);

Me(5,5)=Me(2,2);          Me(6,5)=-Me(3,2);         Me(7,5)=-Me(11,1);
Me(8,5)=Me(11,2);          Me(9,5)=Me(6,2);
Me(10,5)=-Me(8,1);         Me(11,5)=Me(8,2);          Me(12,5)=-Me(9,2);

Me(6,6)=Me(3,3);          Me(7,6)=-Me(12,1);        Me(8,6)=-Me(6,2);
Me(9,6)=Me(12,3);          Me(10,6)=Me(9,1);
Me(11,6)=-Me(9,2);        Me(12,6)=Me(9,3);

Me(7,7)=Me(1,1);          Me(8,7)=-Me(2,1);         Me(9,7)=-Me(3,1);
Me(10,7)=Me(4,1);          Me(11,7)=-Me(5,1);          Me(12,7)=-Me(6,1);

Me(8,8)=Me(2,2);          Me(9,8)=Me(3,2);          Me(10,8)=-Me(4,2);
Me(11,8)=Me(5,2);          Me(12,8)=Me(6,2);

Me(9,9)=Me(3,3);          Me(10,9)=-Me(4,3);        Me(11,9)=Me(5,3);
Me(12,9)=Me(6,3);

Me(10,10)=Me(1,1);        Me(11,10)=-Me(2,1);       Me(12,10)=Me(3,1);

Me(11,11)=Me(2,2);        Me(12,11)=-Me(3,2);

Me(12,12)=Me(3,3);

for i=1:12
   for j=i:12
      Me(i,j)=Me(j,i);
   end
end
%-----------------------------------------------------------------
```

8.2　结构振动控制

8.2.1　线性二次型最优控制

若系统是线性的，性能指标为二次型函数，则最优控制问题称为线性二次型问题。由于二次型性能指标具有鲜明的物理意义，它代表了大量工程实际问题中提出的性能指标要求，并且在数学处理上比较简单，可得到状态线性反馈的最优控制规律，易于构成闭环最

优控制，便于工程实现，因而在实际工程问题中得到了广泛应用.

1. 线性二次型(LQR)经典最优控制

假设线性时不变系统的状态方程模型为

$$\dot{x}(t)=Ax(t)+Bu(t),\quad x(t_0)=x_0 \tag{8-15}$$

设计最优控制控制规律 $u(t)\left(t\in[t_0,t_f]\right)$ 使如下的目标函数 J 达到最小

$$J=\frac{1}{2}x^{\mathrm{T}}(t_f)Sx(t_f)+\frac{1}{2}\int_{t_0}^{t_f}\left\{x^{\mathrm{T}}(t)Q(t)x(t)+u^{\mathrm{T}}(t)R(t)u(t)\right\}\mathrm{d}t \tag{8-16}$$

式中：半正定对称矩阵 S 为对控制系统的终值给出的某种约束；半正定对称矩阵 Q 和正定对称矩阵 R 为对状态变量和输入变量的加权矩阵. 为使问题简化，几个权矩阵通常选为对角阵.

构造哈密顿函数 H 为

$$H(x,u,\lambda)=\frac{1}{2}x^{\mathrm{T}}(t)Q(t)x(t)+\frac{1}{2}u^{\mathrm{T}}(t)R(t)u(t)+\lambda^{\mathrm{T}}(t)\left[Ax(t)+Bu(t)\right] \tag{8-17}$$

由于控制 $u(t)$ 不受约束，则目标函数的最优值可以由求解 H 对 u 的导数所构成的方程得出

$$\frac{\partial H}{\partial u}=R(t)u(t)+B^{\mathrm{T}}\lambda(t)=0 \tag{8-18}$$

由此可以得出最优控制信号 $u(t)$（记之为 $u^*(t)$）如下

$$u^*(t)=-R^{-1}(t)B^{\mathrm{T}}\lambda(t) \tag{8-19}$$

设 $\lambda(t)$ 具有如下的形式

$$\lambda(t)=P(t)x(t) \tag{8-20}$$

式中，$P(t)$ 为对称矩阵，它满足下面著名的 Riccati 微分方程

$$\dot{P}(t)=-P(t)A-A^{\mathrm{T}}P(t)+P(t)BR^{-1}B^{\mathrm{T}}P(t)-Q,\qquad P(t_f)=S \tag{8-21}$$

由于在具有工程实际意义的相当长的时间内，有 $\dot{P}(t)=0$，从而求解 Riccati 微分方程的问题转化为求解如下的 Riccati 代数方程

$$-P(t)A-A^{\mathrm{T}}P(t)+P(t)BR^{-1}B^{\mathrm{T}}P(t)-Q=0 \tag{8-22}$$

则这段时间内的最优控制 $u^*(t)$ 可表示为

$$u^*(t)=-R^{-1}(t)B^{\mathrm{T}}Px(t)=-Kx(t) \tag{8-23}$$

其中 $x(t)$ 满足如下线性齐次方程

$$\dot{x}(t)=(A-BK)x(t) \tag{8-24}$$

以及相应的初始条件：$x(0)=x_0$.

控制系统工具箱中通过 lqr()函数在给定加权矩阵前提下设计 LQR 最优控制器，该函数的调用格式为

```
K=lqr(A, B, Q, R)
[K, P]=lqr(A, B, Q, R)
[K, P, E]=lqr(A, B, Q, R);
```
E 为闭环系统[(A-BK), B, C, D]的特征值.

【例 8.7】 平面桁架结构如图 8.15 所示，两个作动器的作动力分别作用在垂直方向和水平方向. 结构参数为：$E=2.0\times10^{11}\text{N/m}^2$，$\rho=7860\text{kg/m}^3$，$A=0.000\,25\text{m}^2$. 假设初始位移扰动为 $\boldsymbol{x}(0)=[0.1,0,0,0.2,-0.2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]^{\text{T}}$，控制前后节点 2 的水平和垂直位移如图 8.16(a)、(b)所示. 两个作动器的输入电压分别如图 8.17(a)、(b)所示.

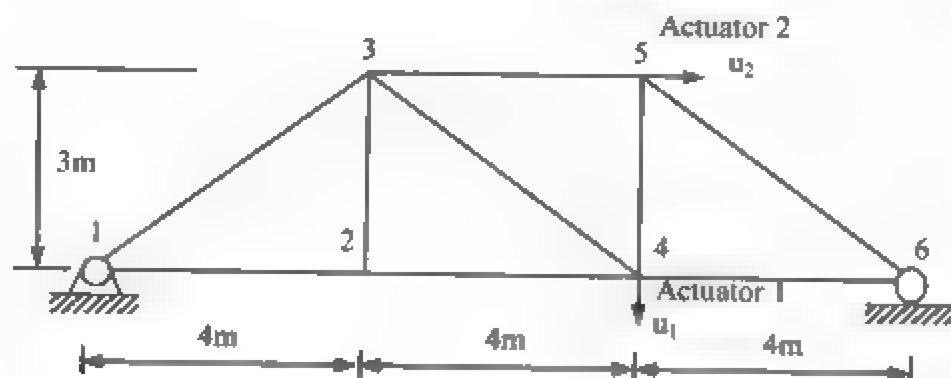
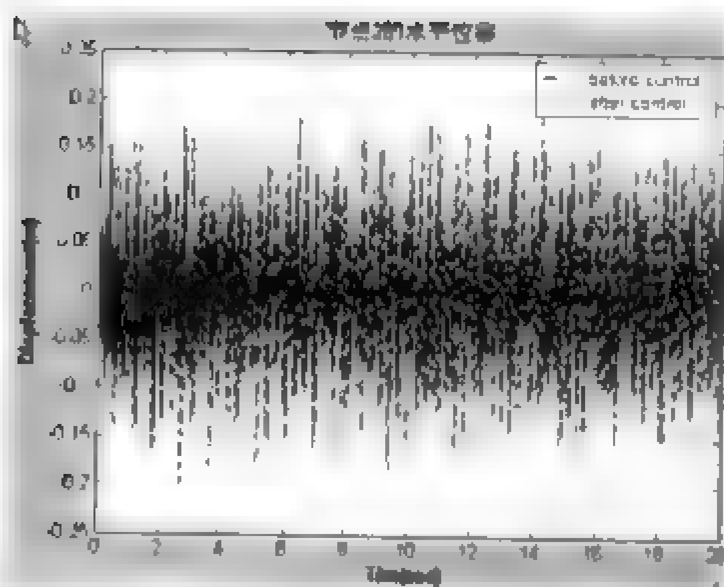
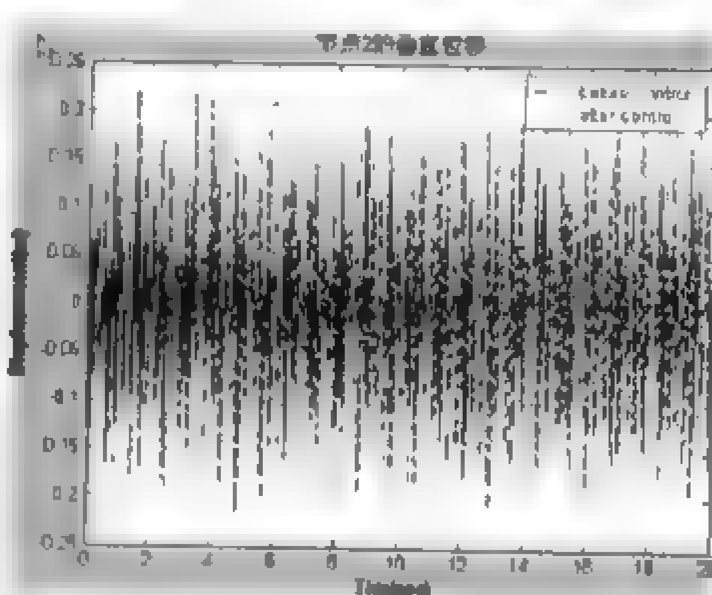


图 8.15　平面桁架结构

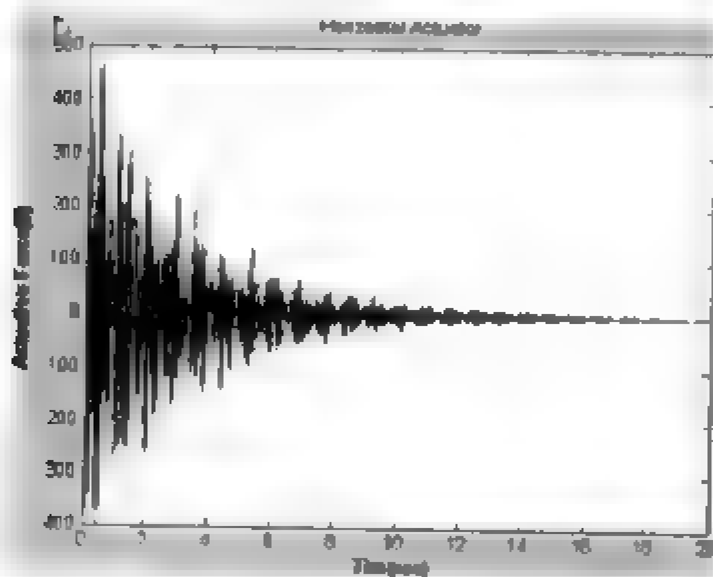


(a) 水平位移

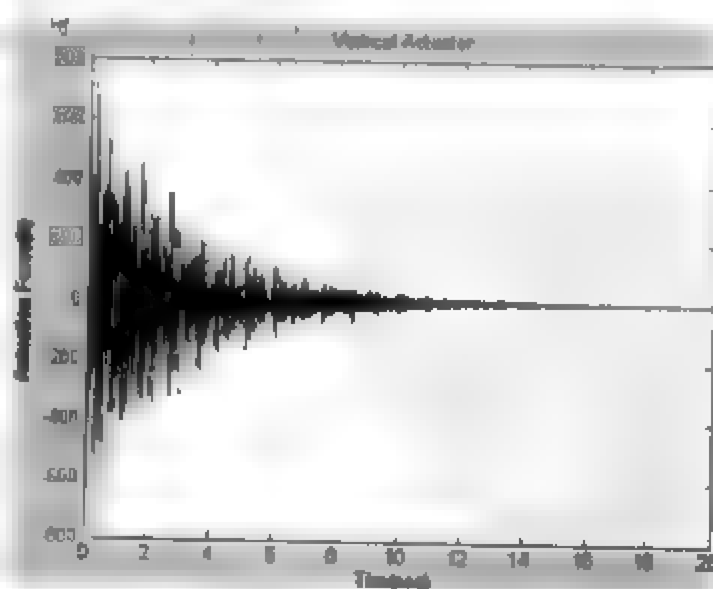


(b) 垂直位移

图 8.16　控制前后节点 2 的水平和垂直位移



(a) 水平布置的作动器



(b) 垂直布置的作动器

图 8.17　作动器的出力曲线

```
%------------------------------------------------------------------------%
% Example 8.7
% to solve LQR design of 2-d truss structure
%
% Problem description
%   Find LQR design of a truss structure  as shown in Fig. 8.6
%
% Variable descriptions
%   k = element stiffness matrix
%   m = element mass matrix
%   kk = system stiffness matrix
%   mm = system mass vector
%   index = a vector containing system dofs associated with each element
%   gcoord = global coordinate matrix
%   prop = element property matrix
%   nodes = nodal connectivity matrix for each element
%   bcdof = a vector containing dofs associated with boundary conditions
%   bcval = a vector containing boundary condition values associated with
%          the dofs in 'bcdof'
%------------------------------------------------------------------------%

%---------------------------
%  control input data
%---------------------------

nel=9;          % number of elements
nnel=2;         % number of nodes per element
ndof=2;         % number of dofs per node
nnode=6;        % total number of nodes in system
sdof=nnode*ndof; % total system dofs

%-------------------------
%  nodal coordinates
%-------------------------
gcoord(1,1)=0.0;  gcoord(1,2)=0.0;
gcoord(2,1)=4.0;  gcoord(2,2)=0.0;
gcoord(3,1)=4.0;  gcoord(3,2)=3.0;
gcoord(4,1)=8.0;  gcoord(4,2)=0.0;
gcoord(5,1)=8.0;  gcoord(5,2)=3.0;
gcoord(6,1)=12.0;  gcoord(6,2)=0.0;

%-------------------------------------------
%  material and geometric properties
%-------------------------------------------

prop(1) 200e9;        % elastic modulus
prop(2)=0.0025;       % cross-sectional area
prop(3)=7860;         % density
```

```
%  -----  -------  -----
% nodal connectivity
%  -----  -------  ------

nodes(1,1)=1;  nodes(1,2) 2;
nodes(2,1)=1;  nodes(2,2)=3;
nodes(3,1)=2;  nodes(3,2)=3;
nodes(4,1)=2;  nodes(4,2)-4;
nodes(5,1)=3;  nodes(5,2)=4;
nodes(6,1)=3;  nodes(6,2)=5;
nodes(7,1)=4;  nodes(7,2)=5;
nodes(8,1)=4;  nodes(8,2)-6;
nodes(9,1)=5;  nodes(9,2)=6;

%---  -----  ------------------
% applied constraints
%--------------------------

bcdof(1)=1;     % 1st dof (horizontal displ) is constrained
bcval(1)=0;     % whose described value is 0
bcdof(2)=2;     % 2nd dof (vertical displ) is constrained
bcval(2)=0;     % whose described value is 0
bcdof(3)=12;    % 12th dof (horizontal displ) is constrained
bcval(3)=0;     % whose described value is 0

%--------------------------
% initialization to zero
%--------------------------

kk=zeros(sdof,sdof);          % system stiffness matrix
mm=zeros(sdof,sdof);          % system mass matrix
index=zeros(nnel*ndof,1);     % index vector

%--------------------------
% loop for elements
%--------------------------

for iel=1:nel   % loop for the total number of elements

nd(1)-nodes(iel,1);   % 1st connected node for the (iel)-th element
nd(2)=nodes(iel,2);   % 2nd connected node for the (iel)-th element

x1=gcoord(nd(1),1); y1=gcoord(nd(1),2);  % coordinate of 1st node
x2=gcoord(nd(2),1); y2=gcoord(nd(2),2);  % coordinate of 2nd node

leng=sqrt((x2-x1)^2+(y2-y1)^2);  % element length

if (x2-x1)  0;
```

```
beta=2*atan(1);      % angle between local and global axes
else
beta=atan((y2-y1)/(x2-x1));
end

el=prop(1);          % extract elastic modulus
area=prop(2);        % extract cross-sectional area
rho=prop(3);         % extract mass density

index=feeldof(nd,nnel,ndof);  % extract system dofs for the element

ipt=1;                        % flag for consistent mass matrix
[k,m]=fetruss2(el,leng,area,rho,beta,ipt); % element matrix

kk=feasmbl1(kk,k,index);       % assemble system stiffness matrix
mm=feasmbl1(mm,m,index);       % assemble system mass matrix

end

%------------------------------------------
%  apply constraints
%------------------------------------------

[kk,mm]=feaplycs3(kk,mm,bcdof);  % apply the boundary conditions

%----------------------------------------------------------------------
% transform into the first order state space form equation
%----------------------------------------------------------------------
sodfm=sodf-length(bcdof);
F=zeros(sdofm,2);
F(6,1)=1;
F(7,2)=1;
A=[zeros(sdofm), eye(sdofm); -inv(mmm)*kkm, zeros(sodfm)];
B=[zeros(sdofm, 2); inv(mmm)*F];
C=[1 0 0 0 0 0 0 0 0];
D=0;
%-------------------------------------------------------------------
% design the LQR control law
%-------------------------------------------------------------------
Q=eye(sodfm);
R=0.00001*eye(2);
[K, P]=lqr(A, B, Q, R);
%----------------------------------------------------------------------
% response comparison for the open-loop system and the close-loop system
%----------------------------------------------------------------------
X0=[0.1, 0, 0, 0.2, -0.2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
G=ss(A, B, C, D);
t=0:0.01:10;
[y, t, x]=initial(G, X0, t);
```

```
Gm=ss((A-B*K), B, C, D);
[ym, t, xm]=initial(Gm, X0, t);
u=-inv(R)*B'*P*xm;
plot(t, x(:,1),'-.',t, xm(:,1))
legend('before control','after control')
xlabel('Tim(sec)','FontWeight','bold')
ylabel('Displacement(m)','FontWeight','bold')
title('节点2的水平位移')
figure
plot(t,x(:,2),'-.',t,xm(:,2))
legend('before control','after control')
xlabel('Tim(sec)','FontWeight','bold')
ylabel('Displacement(m)','FontWeight','bold')
title('节点2的垂直位移')
figure
plot(t,u(1,:))
xlabel('Tim(sec)','FontWeight','bold')
ylabel('Actuation Force(N)','FontWeight','bold')
title('Horizontal Actuator')
figure
plot(t,u(2,:))
xlabel('Tim(sec)','FontWeight','bold')
ylabel('Actuation Force(N)','FontWeight','bold')
title('Vertical Actuator')
%
%-----------------------------------------------------------------------


function [kkm, mmm]=feaplycs3(kk, mm, bcdof)
%-----------------------------------------------------------------------
% Purpose:
% Apply constraints to obtain the final system mass matrix and stiffness matrix
%
% Synopsis:
%     [kkm,mm]=feaplycs(kk,mm,bcdof)
%
% Variable Description:
%   kkm- system stiffness matrix after applying constraints
%   mmm- system mass matrix after applying constraints
%    kk - system stiffness matrix before applying constraints
%    mm - system mass matrix before applying constraints
%    bcdof - a vector containging constrained d.o.f
%    ----  ----  ----  ----  ----  ---------  ----  ----  ---  ---  ---
bcdof=sort(bcdof);
n=length(bcdof);
j=-1
for i=1:n
j=j+1;
mm(:, (bcdof(i)-j))=[];
```

```
mm((bcdof(i)-j), :)=[];
kk(:, (bcdof(i)-j))=[];
kk((bcdof(i) j), :)=[];
end
kkm=kk;
mmm mm;
%-          --------            ---------
```

2. 线性二次型Gauss(LQG)最优控制

设受控线性定常系统的状态方程为

$$\dot{x}(t)=Ax(t)+Bu(t)+\varepsilon_1(t)\quad x(t_0)=x_0 \tag{8-25}$$

$$y(t)=Cx(t)+\varepsilon_2(t) \tag{8-26}$$

式中，$\varepsilon_1(t)$和$\varepsilon_2(t)$为白噪声信号，分别为对状态变量量测与输出量测的随机扰动. 假设(A,C)是可观的，$\varepsilon_1(t)$和$\varepsilon_2(t)$均为零均值Gauss白噪声，且有

$$\left.\begin{array}{ll} E\left[\varepsilon_1(t)\right]=0 \quad E\left[\varepsilon_2(t)\right]=0 & \\ E\left[\varepsilon_1(t)\,\varepsilon_1^{\mathrm{T}}(\tau)\right]=Q_e\delta(t-\tau) & Q_e=Q_e^{\mathrm{T}}\geqslant 0 \\ E\left[\varepsilon_2(t)\,\varepsilon_2^{\mathrm{T}}(\tau)\right]=R_e\delta(t-\tau) & R_e=R_e^{\mathrm{T}}\geqslant 0 \end{array}\right\} \tag{8-27}$$

式中，$E[\bullet]$是均值.

根据 **LQG 问题的分离原理**，首先采用 LQR 最优控制算法设计全状态反馈最优控制力$u(t)$，即

$$u(t)=-Kx(t) \tag{8-28}$$

然后根据结构的观测输出，采用 Kalman 滤波器估计结构的全部状态. 为此，选取如下目标函数

$$J_e=E\left[\{x(t)-\hat{x}(t)\}^{\mathrm{T}}\{x(t)-\hat{x}(t)\}\right] \tag{8-29}$$

式中，$\hat{x}(t)$是状态$x(t)$的估计. 由此构造 Kalman 滤波器为

$$\dot{\hat{x}}=A\hat{x}+Bu+K_e(y-\hat{y})\qquad \hat{x}(t_0)=\hat{x}_0 \tag{8-30}$$

$$\hat{y}=C\hat{x} \tag{8-31}$$

式中，Kalman 滤波器的增益矩阵K_e可以由下式得出

$$K_e=P_eC^{\mathrm{T}}R_e^{-1} \tag{8-32}$$

式中，P_e满足下面的 Riccati 代数方程

$$P_eA^{\mathrm{T}}+AP_e-P_eC^{\mathrm{T}}R_e^{-1}CP_e+Q_e=0 \tag{8-33}$$

最后，系统的最优控制可以表示为状态估计的反馈

$$u(t)=-K\hat{x}(t) \tag{8-34}$$

将式(8-34)代入式(8-25)得到受控系统的状态方程为

$$\dot{\hat{x}}=(A-BG-K_eC)\hat{x}+K_ey\quad \hat{x}(t_0)=\hat{x}_0 \tag{8-35}$$

$$\hat{y}=C\hat{x} \tag{8-36}$$

【例 8.8】 三层剪切型框架结构如图 8.18 所示，设结构层间质量和层间刚度分别为

$m_i=4\times10^5$kg 和 $k_i=2\times10^8$N/m ($i=1,2,3$). 结构阻尼矩阵 $\boldsymbol{C}=0.7334\boldsymbol{M}+0.0026\boldsymbol{K}$. 结构的外干扰为 EI Centro (NS, 1940)地震波，地震输入峰值为200Gal， 地震作用位置矩阵 $\boldsymbol{D}_s=-\boldsymbol{M}\{1\}$ ， 其中 $\{1\}$ 是元素均为 1 的列向量. 假设 $Q_e=E\left[\varepsilon_1(t)\,\varepsilon_1^{\mathrm{T}}(\tau)\right]=10^{-4}$ ，$\boldsymbol{R}_e=E\left[\varepsilon_2(t)\,\varepsilon_2^{\mathrm{T}}(\tau)\right]=10^{-2}\boldsymbol{I}$. 控制力位置矩阵为 $\boldsymbol{B}_s=\begin{bmatrix}1&-1&0\\0&1&-1\\0&0&1\end{bmatrix}$. 分别采用 LQG 和 LQR 方法进行控制，第一层位移的计算比较如图 8.19 所示.

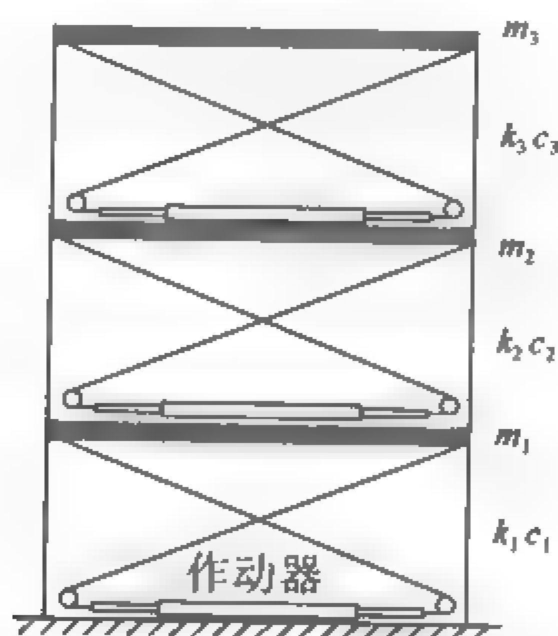


图 8.18 三层剪切型框架结构

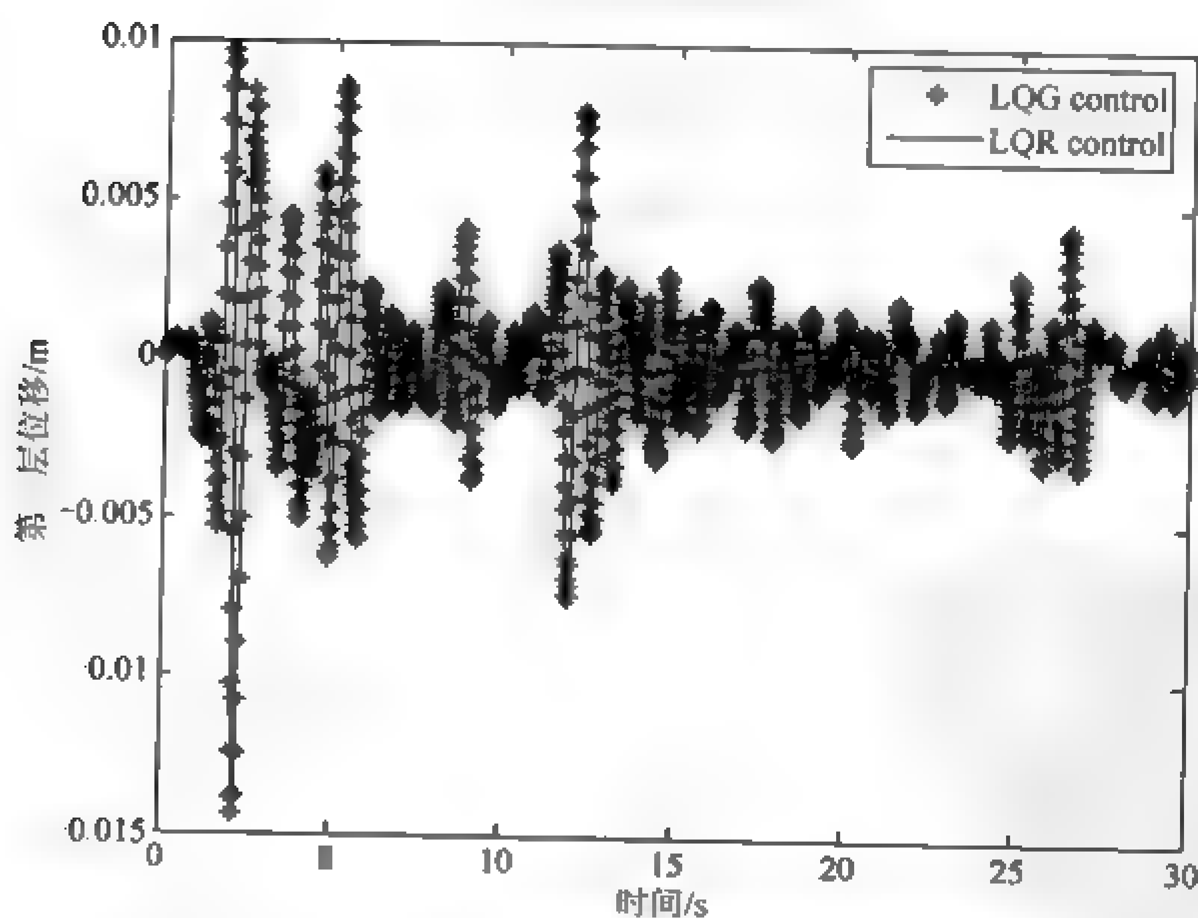


图 8.19 采用 LQG 和 LQR 方法进行控制的第一层位移的计算比较

M 文件如下：

```
%-------------------------------------------------------------------
% Ex.8.8
% to execute LQG and LQR designs of frame structure
%-------------------------------------------------------------------
M=4e5*eye(3); k1=2e8; k2=k1; k3=k2;
K=[k1+k2, -k2, 0; -k2, k2+k3, -k3; 0, -k3, k3];
C=0.7334*M+0.0026*K;
Bs=[1, -1, 0; 0, 1, -1; 0, 0, 1];
Ds=-M*[1;1;1];
A=[zeros(3), eye(3); -inv(M)*K, -inv(M)*C];
B=[zeros(3); inv(M)*Bs];
D=[zeros(3,1);inv(M)*Ds];
Q=100*[K, zeros(3); zeros(3), M];
R=5e-6*eye(3);
G=lqr(A, B, Q, R);
C0=[0 0 0 1 0 0;0 0 0 0 1 0;0 0 0 0 0 1];
Qe=1e-4;
Re=1e-2*eye(3);
Ke=lqe2(A, D, C0, Qe, Re);
Am=A-B*G-Ke*C0*A+Ke*C0*B*G;
Amm=inv(eye(6)-Ke*C0)*Am;
Bmm=D;
load BO.txt.dat
Xg=BO(:,2)
Xg=Xg(1:1501);
Xg=Xg/max(Xg)*2;
X0=zeros(6,1);
C1=eye(6);
D0=0;
G1=ss(Amm,Bmm, C1, D0);
t=0:0.02:30;
[y, t, x]=lsim(G1, Xg, t, X0);
G2=ss(A-B*G,Bmm,C1,D0);
[y1, t1, x1]=lsim(G2, Xg, t, X0);
hold on
plot(t, x(:,1), '*')
plot(t, x1(:,1))
hold off
legend('LQG control', 'LQR control')
xlabel('时间/s')
ylabel('第一层位移/m')
%-------------------------------------------------------------------
```

8.2.2　线性定常系统的极点配置

通过采用极点配置的方法使系统的闭环极点配置在所期望的位置上，从而达到一定的

性能指标要求. 由于状态反馈不能改变系统的不可控模态，因此系统通过状态反馈可以任意配置闭环极点的必要条件为受控系统完全可控.

控制系统工具箱提供了基于鲁棒极点配置算法的 place()函数，用来求取状态反馈矩阵，其调用格式为

K = place(A,B,P)

函数中参数 P 为包含期望极点位置的向量，返回的变量 K 为状态反馈向量.

【例 8.9】 对例 8.8 的结构采用极点配置进行控制，要求闭环系统特征值配置为：$[-1.8218\pm9.9378i \quad -5.1043\pm27.8451i \quad 7.5273\pm40.2094i]$，则控制前后第一层位移曲线如图 8.20 所示.

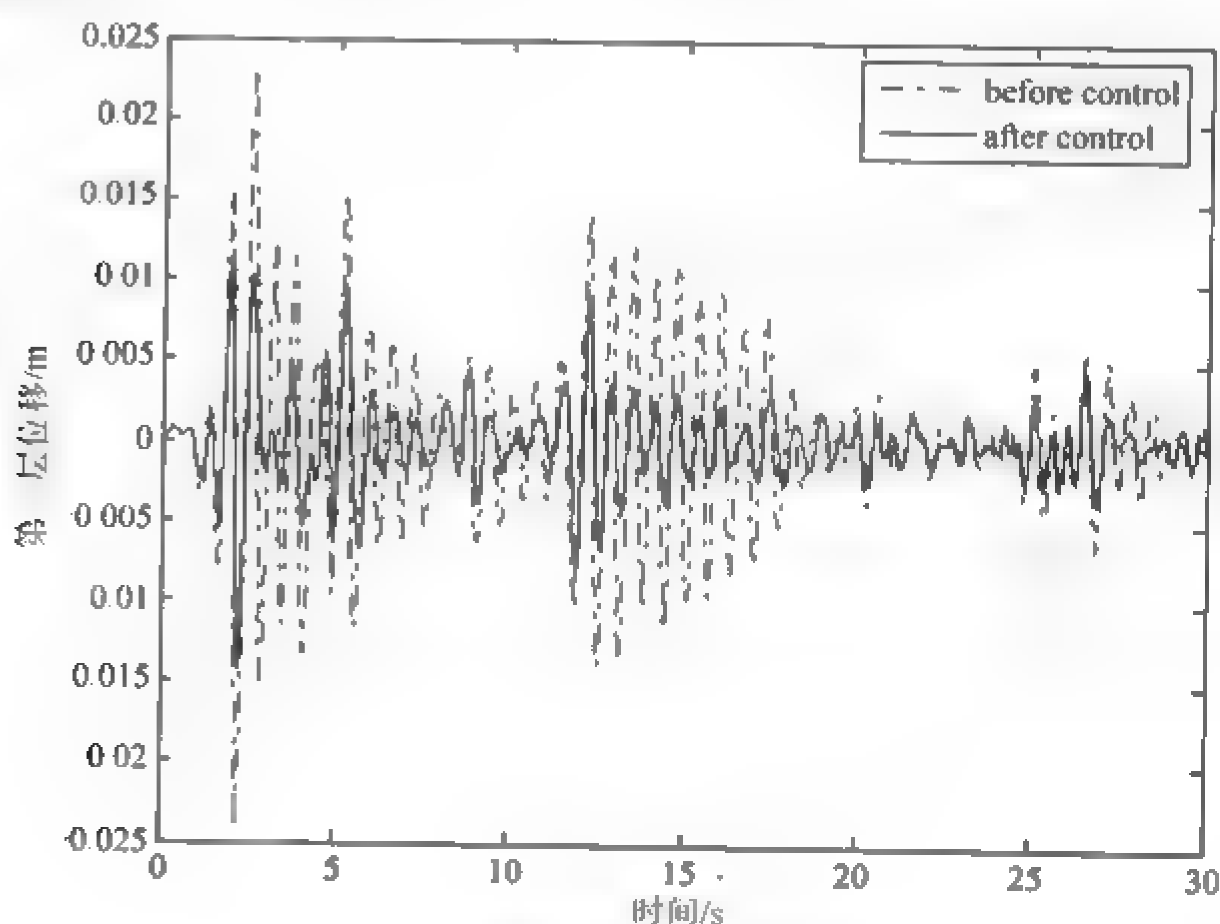


图 8.20 控制前后的第一层位移曲线

M 文件如下：

```
%------------------------------------------------------------------
% Ex 8.9
% to execute pole assignment design of frame structure
%------------------------------------------------------------------

M=4e5*eye(3); k1=2e8; k2=k1; k3=k2;
K=[k1+k2, -k2, 0; -k2, k2+k3, -k3; 0, -k3, k3];
C=0.7334*M+0.0026*K;
Bs=[1, -1, 0; 0, 1, -1; 0, 0, 1];
Ds=-M*[1;1;1];
A=[zeros(3), eye(3); -inv(M)*K, -inv(M)*C];
B=[zeros(3); inv(M)*Bs];
C0=eye(6);
```

```
D0 0;
D=[zeros(3,1);inv(M)*Ds];
lambda1=[-1.8218+9.9378*i, -1.8218-9.9378*i, -5.1045+27.8451*i,...
-5.1045-27.8451*i  -7.5273+40.2094*i, -7.5273-40.2094*i];
Ph1=inv(lambda1(1)*eye(6)-A)*B;
Ph2=inv(lambda1(2)*eye(6)-A)*B;
Ph3=inv(lambda1(3)*eye(6)-A)*B;
Ph4=inv(lambda1(4)*eye(6)-A)*B;
Ph5=inv(lambda1(5)*eye(6)-A)*B;
Ph6=inv(lambda1(6)*eye(6)-A)*B;
e=[1, 1, 0, 0, 0, 0; 0, 0,-1, 1, 0, 0; 0, 0, 0, 0, 1, 1];
Ke=-e*inv([Ph1(:,1)  Ph2(:,1)  Ph3(:,2)  Ph4(:,2)  Ph5(:,3)  Ph6(:,3)]);
Ke=real(Ke);
load BO.txt
xg=BO(:,2)
xg=xg(1:1501)
xg=xg/(max(xg))*2;
X0=zeros(6,1);
G=ss(A,D, C0, D0);
G1=ss(A-B*Ke, D, C0, D0);
t=0:0.01:15;
[y, t, x]=lsim(G, xg, t, X0);
[y1, t, x1]=lsim(G1, xg, t, X0);
plot(t, x(:,1), '-.', t, x1(:,1))
legend('before control', 'after control')
xlabel('时间/s')
ylabel('第一层位移/cm',)
%-----------------------------------------------------------------------
```

8.2.3 线性定常系统的模态控制

由振动理论可知，系统或结构的振动可以将它置于模态空间来考察，其在时间域内的振动通常可以用其在模态空间内的前几阶模态的振动近似描述. 这样，结构在时间域内的振动控制可转化为在模态空间内少数几个模态的振动控制，这就是振型(也即模态)控制，模态控制的方法可分为模态耦合空间控制法和独立模态空间控制(IMSC)法.

对于结构振动问题，直接由结构运动方程得到振型(模态)更为方便，也具有更明确的物理意义. 因此，按结构动力学的方法将运动方程转化为广义坐标方程，然后由广义坐标方程写出状态方程，即可按上述方法确定最优控制力向量.

考虑以下n自由度受控系统的运动方程：

$$\boldsymbol{M}\ddot{\boldsymbol{q}}(t)+\boldsymbol{C}\dot{\boldsymbol{q}}(t)+\boldsymbol{K}\boldsymbol{q}(t)=\boldsymbol{B}\boldsymbol{u}(t) \tag{8-37}$$

$$\boldsymbol{q}(t_0)=\boldsymbol{q}_0 \qquad \dot{\boldsymbol{q}}(t_0)=\dot{\boldsymbol{q}}_0 \tag{8-38}$$

设该系统无阻尼模态(振型)矩阵为$\boldsymbol{\Phi}$， 作模态变换

$$\boldsymbol{q}(t)=\boldsymbol{\Phi}z(t) \tag{8-39}$$

式中，$z(t)=\left[z_1(t) \quad z_2(t) \quad \cdots \quad z_n(t)\right]^{\mathrm{T}}$是系统的广义模态坐标向量. 将式(8-39)代入方程式(8-37)，然后左乘$\boldsymbol{\Phi}^{\mathrm{T}}$，并假定阻尼矩阵$\boldsymbol{C}$关于模态矩阵$\boldsymbol{\Phi}$正交，则得广义模态坐标运动方程

$$\boldsymbol{M}^*\ddot{\boldsymbol{q}}(t)+\boldsymbol{C}^*\dot{\boldsymbol{q}}(t)+\boldsymbol{K}^*\boldsymbol{q}(t)=\boldsymbol{U}^*(t) \tag{8-40}$$

其中

$$\left.\begin{aligned} \boldsymbol{M}^* &= \mathrm{diag}\left[M_i^*\right]=\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{\Phi} \\ \boldsymbol{C}^* &= \mathrm{diag}\left[C_i^*\right]=\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{\Phi} \\ \boldsymbol{K}^* &= \mathrm{diag}\left[K_i^*\right]=\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\Phi} \\ \boldsymbol{U}^*(t) &= \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{B}\boldsymbol{u}(t)=\boldsymbol{L}\boldsymbol{u}(t) \end{aligned}\right\} \tag{8-41}$$

式中，$\boldsymbol{L}=\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{B}$为$n\times p$维矩阵，$p$是控制器数目.

假定仅考虑结构系统的n_c个广义模态坐标($n_c\leqslant n$)，并在相应的矩阵和向量下用下标c标记，则其相应的广义模态坐标运动方程为

$$\boldsymbol{M}_c^*\ddot{\boldsymbol{q}}_c(t)+\boldsymbol{C}_c^*\dot{\boldsymbol{q}}_c(t)+\boldsymbol{K}_c^*\boldsymbol{q}_c(t)=\boldsymbol{U}_c^*(t) \tag{8-42}$$

从式(8-42)出发，总可以由某种控制算法求得最优控制力向量

$$\boldsymbol{U}_c^*(t)=\left[U_{c1}^*(t) \quad U_{c2}^*(t) \quad \cdots \quad U_{cn_c}^*(t)\right],$$

并可表示为

$$\boldsymbol{U}_c^*(t)=-\boldsymbol{G}_c\begin{Bmatrix}\boldsymbol{q}_c(t)\\ \dot{\boldsymbol{q}}_c(t)\end{Bmatrix}=-\boldsymbol{G}_{c1}\boldsymbol{q}_c(t)-\boldsymbol{G}_{c2}\dot{\boldsymbol{q}}_c(t) \tag{8-43}$$

式中，$\boldsymbol{G}_c=\left[\boldsymbol{G}_{c1} \quad \boldsymbol{G}_{c2}\right]$为相应的$n_c\times 2n_c$维模态增益反馈矩阵.

【例 8.10】 耦合模态空间控制：空间 36 杆智能桁架结构如图 8.5 所示，普通构件的质量密度和杨氏弹性模量分别是$2710\,\mathrm{kg/m^3}$和$70\mathrm{GPa}$，横截面面积为$0.0025\mathrm{m}^2$. 压电主动构件的两连接杆的长度相同，其质量密度、杨氏模量和横截面面积分别是$8000\mathrm{kg/m^3}$、$210\mathrm{GPa}$和$6.9\times10^{-5}\mathrm{m}^2$. 压电堆的质量密度、杨氏模量和横截面面积为$7600\mathrm{kg/m^3}$、$63\mathrm{GPa}$和$7.07\times10^{-4}\mathrm{m}^2$，由 490 块圆形压电薄片叠成，相应有效长度为$0.56\mathrm{m}$，等效的应力系数$e_{33}^s$为$1746\mathrm{N/(V\cdot m)}$. 初始扰动矢量等效于在节点 12 的$y$方向施加一大小为 10 000 N 的力产生的瞬态位移. 3 个作动器分别配置在杆件标号为 7、24 和 26 的位置，以控制结构振动的前 6 阶模态. 控制器设计变量$R=1\times10^{-3}[0.7987 \quad 0.8875 \quad 0.7917]$. 控制前后节点 12 的$x$方向位移曲线如图 8.21 所示. 3 个作动器的输出电压如图 8.22(a)~(c)所示.

```
%-----------------------------------------------------------------------
% Example 8.10
% optimal control design in the coupling(independent)modal space for
  space 36-bar piezoelectric
% intelligent truss
% Problem description
% the modal control of a 36 bar space intelligent truss as shown in Fig. 8.6
%
```

```
% Variable descriptions
%   k = element stiffness matrix
%   m = element mass matrix
%   kk = system stiffness matrix
%   mm = system mass vector
%   index = a vector containing system dofs associated with each element
%   gcoord = global coordinate matrix
%   nodes = nodal connectivity matrix for each element
%   bcdof = a vector containing dofs associated with boundary conditions
%   bcval - a vector containing boundary condition values associated with
%           the dofs in 'bcdof'
%-----------------------------------------------------------------------
D =[7   24  26];                % the actuator positions
R = 1.0e-003 *[0.7987   0.8875   0.7917]; % the control gains

%---------------------------
% initialize structural parameters
%-----------------------------------------
nel=36;          % number of elements
nnel=2;          % number of nodes per element
ndof=3;          % number of dofs per node
nnode=12;        % total number of nodes in system
sdof=nnode*ndof; % total system dofs

%-------------------------
%  nodal coordinates
%-------------------------
gcoord=[0 0 0;1 0 0;1 1 0;0 1 0;...
     0 0 1;1 0 1;1 1 1;0 1 1;...
     0 0 2;1 0 2;1 1 2;0 1 2];
%---------------------------
% nodal connectivity
%---------------------------
nodes=zeros(36,2);
nodes(1,1)=1;    nodes(1,2)=5;
nodes(2,1)=2;    nodes(2,2)=6;
nodes(3,1)=3;    nodes(3,2)=7;
nodes(4,1)=4;    nodes(4,2)=8;
nodes(5,1)=5;    nodes(5,2)=6;
nodes(6,1)=6;    nodes(6,2)=7;
nodes(7,1)=7;    nodes(7,2)=8;
nodes(8,1)=5;    nodes(8,2)=8;
nodes(9,1)=6;    nodes(9,2)=8;
nodes(10,1)=5;   nodes(10,2)=7;
nodes(11,1)=1;   nodes(11,2)=8;
```

```
nodes(12,1)=4;   nodes(12,2)=5;
nodes(13,1)=4;   nodes(13,2)=7;
nodes(14,1)=3;   nodes(14,2)=8;
nodes(15,1)=3;   nodes(15,2)=6;
nodes(16,1)=2;   nodes(16,2)=7;
nodes(17,1)=2;   nodes(17,2)=5;
nodes(18,1)=1;   nodes(18,2)=6;

for i=1:1
   for j=1:18
      nodes(i*18+j,1)=nodes(j,1)+4*i;
      nodes(i*18+j,2)=nodes(j,2)+4*i;
   end
end

%--------  ---------  ------
% applied constraints
%---------    --------  ----
for i=1:12
   bcdof(i)=i;
   bcval(i)=0;
end
ncon=length(bcdof);     % the number of constraint degrees of freedom
%-----   ---------- --------
% initialization to zero
%-------  ---------  --------
leng=zeros(nel,1);   % element length
beta=zeros(nel,1);   % angle between local and global axes
ff=zeros(sdof,1);    % system force vector for initial disturbance
kk=zeros(sdof,sdof);          % system stiffness matrix
mm=zeros(sdof,sdof);         % system mass matrix
index=zeros(nnel*ndof,1);      % index vector
k=zeros(nnel*ndof,nnel*ndof);  % element stiffness matrix
m=zeros(nnel*ndof,nnel*ndof);  % element mass matrix
B0=zeros(sdof,nel);
% total possible assigned matrix for piezoelectric actuators
B=zeros((sdof-ncon),nact);     % the actuator distributing matrix
%----------------------------
% applied nodal force
%---     ---    -----    ---

ff(35,1)=10000;         % 12th node has 10000 N in y- direction
%----------------------------
% loop for elements
```

```
%--      ------        -------
for iel=1:nel    % loop for the total number of elements
   nd(1) nodes(iel,1);   % 1st connected node for the (iel)-th element
   nd(2) nodes(iel,2);   % 2nd connected node for the (iel)-th element

   x1=gcoord(nd(1),1); y1=gcoord(nd(1),2); z1=gcoord(nd(1),3);
   % coordinate of 1st node
   x2=gcoord(nd(2),1); y2=gcoord(nd(2),2); z2=gcoord(nd(2),3);
   % coordinate of 2nd node

   leng(iel)=sqrt((x2-x1)^2+(y2-y1)^2+(z2-z1)^2);  % the length for
   (iel)-th element

   c(iel)=sqrt((x2-x1)^2)/leng(iel);
   s(iel)=sqrt((y2-y1)^2)/leng(iel);
   r(iel)=sqrt((z2-z1)^2)/leng(iel);

  for i=1:length(D)
    if D(i)==iel

    [k,m]=fetruss31(leng(iel),c(iel),s(iel),r(iel));
    % compute element stiffness and mass matrix of piezoelectric bar
      else
        el=7.2e10;                          %  elastic modulus
        area=0.0025;                        %  cross-sectional area
        rho=2710;                          %  mass density
       [k,m]=fetruss3(el,leng(iel),area,rho,c(iel),s(iel),r(iel));

    end
  end

index=feeldof(nd,nnel,ndof);  % extract system dofs for the element

kk=feasmbl1(kk,k,index);      % assemble into system stiffness matrix
mm=feasmbl1(mm,m,index);      % assemble into system mass matrix

end
%-----------          -  ----------------------------------------
%  apply constraints and solve the matrix for initial disturbance
%-----------------

[kk,ff]=feaplyc2(kk,ff,bcdof,bcval);    % apply the boundary conditions
disp=kk\ff;                          % solve the matrix equation to find nodal
displacements
disp=disp(13:sdof);
```

```
% --------------------------------------------------------------
% apply optimal control in modal space
% --------------------------------------------------------------
kk1=kk(13:sdof,13:sdof);
mm1=mm(13:sdof,13:sdof);
[V,D1]=eig(kk1,mm1);           % compute eigenvalue and eigenvector
[lambda,k]=sort(diag(real((D1))));
V=real(V(:,k));
Factor=diag(V'*mm1*V);
Vnorm=V*inv(sqrt(diag(Factor)));        % normalize eigenvector
natfreq=sqrt(lambda)/(2*pi);
nt=inv(Vnorm)*disp;
nt=real(nt);

%-------------------------------------------------------------------------
% determine the actuator distributing matrix,including two step:
%  first step: to determine system distributing matrix
%  second step: extract actuator distributing matrix
%-------------------------------------------------------------------------
% first step
% --------------
   for i=1:nact
     if rem(leng(D(i)),sqrt(2))==1
       B0(3*nodes(D(i),1)-2,D(i))=-0.3875588*c(D(i));
       B0(3*nodes(D(i),1)-1,D(i))=-0.3875588*s(D(i));
       B0(3*nodes(D(i),1),D(i))=-0.3875588*r(D(i));
       B0(3*nodes(D(i),2)-2,D(i))=-0.3875588*c(D(i));
       B0(3*nodes(D(i),2)-1,D(i))=-0.3875588*s(D(i));
       B0(3*nodes(D(i),2),D(i))=-0.3875588*r(D(i));

     else
       B0(3*nodes(D(i),1)-2,D(i))=-0.6454421*c(D(i));
       B0(3*nodes(D(i),1)-1,D(i))=-0.6454421*s(D(i));
       B0(3*nodes(D(i),1),D(i))=-0.6454421*r(D(i));
       B0(3*nodes(D(i),2)-2,D(i))=-0.6454421*c(D(i));
       B0(3*nodes(D(i),2)-1,D(i))=-0.6454421*s(D(i));
       B0(3*nodes(D(i),2),D(i))=-0.6454421*r(D(i));

     end
   end
%-------------
% second step
% -----------

for i=1:nact
```

```
for j=1:nel
   if D(i)==j
      for k=1:(sdof-ncon)
         B(k,i)=B0((k+ncon),j);
      end
   end
end
end
%--------------------------------------------------------------------------
% compute modal control force and nodal displacements
% based on couple mode control
%--------------------------------------------------------------------------
ncm=6;                          % the number of controlled modes
A=[zeros(ncm) eye(ncm)
   -diag(lambda(1:ncm)) zeros(ncm)];
B1=[zeros(ncm,nact)
   Vnorm(:,1:ncm)'*B];
%--------------------------------------------------------------------------
% judge whether the system is controlled
%--------------------------------------------------------------------------
indexcon=ctrb(A, B1);
rankc=rank(indexcon);

   t=0:0.001:1;
   n=1/0.001+1;
   x=zeros((sdof-ncon),n);
   x1=zeros((sdof-ncon),n);
   C=eye(2*ncm);
   D2=0;
%--------------------------------------------------------------------------
% the initial values of the modal displacements and velocities
%--------------------------------------------------------------------------

  for i=1:ncm
     q(i,1)=nt(i);
  end
  for i=(ncm+1):(2*ncm)
     q(i,1)=0.0;
  end

     yb=initial(ss(A,B1,C,D2),q,t);    % the modal state vector before control
  for i=1:ncm
     Q(i,1)=real((lambda(i)));
  end
  for i=(ncm+1):(2*ncm)
     Q(i,1)=1;
  end
```

```
    Q=diag(Q);
   R1-diag(R);
   [kc,P]=lqr(A,B1,Q,R1);
   A1-A-B1*kc;
   yc-initial(ss(A1,B1,C,D2),q,t);
   u= inv(R1)*B1'*P*yc';                                   % applied volatges

   %-------------------------------------------------------------------------
   % compute displacement before and control only considering controlled modes
   %-------------------------------------------------------------------------

   x-Vnorm(:,1:ncm)*yb(:,1:ncm)'; % compute nodal displacement before control
   x1=Vnorm(:,1:ncm)*yc(:,1:ncm)'; % compute nodal displacement after control
   %
%---------------------------------------------------------------------------
plot(t,x(23,:),'-.',t, x1(23,:))
legend('before control','after control')
xlabel('Tim(sec)','FontWeight','bold')
ylabel('Displacement(m)','FontWeight','bold')
title('节点12的y方向位移')
figure
plot(t,u(1,:))
xlabel('Tim(sec)','FontWeight','bold')
ylabel('Actuation Voltage(V)','FontWeight','bold')
title('杆号7的作动器')
figure
plot(t,u(2,:))
xlabel('Tim(sec)','FontWeight','bold')
ylabel('Actuation Voltage(V)','FontWeight','bold')
title('杆号24的作动器')
figure
plot(t,u(3,:))
xlabel('Tim(sec)','FontWeight','bold')
ylabel('Actuation Voltage(V)','FontWeight','bold')
title('杆号26的作动器')
%
%---------------------------------------------------------------------------
 function [k, m] fetruss31(leng,c,s,r)

%  ------------------------------------------------------------------------
%  Purpose:
%     Stiffness and mass matrices for piezpelectric element
%
%  Synopsis:
%     [k,m]-fetruss31(leng,c,s,r)
%
%  Variable Description:
```

```
%     k - element stiffness matrix (size of 6x6)
%     m - element mass matrix (size of 6x6)
%    leng - element length
%     c,s,r - direction cosine between global coordinate and local coordinate
%
%   One assumption: the lengths of two connected bars are same
%
%-----------------------------------------------------------------------
 el1=210e9;                   % Young's module of connected bars
 rho1=8000;                   % mass density of connected bars
 area1=6.9e-5;                % cross-section area of connected bars
 el2=63e9;                    % Young's module of piezoelectric stack
 rho2=7600;                   % mass density of piezoelectric stack
 area2=7.07e-4;               % cross-section area of piezoelectric stack
 leng2=0.56;
 k1=area1*el1/((leng-leng2)/2);
 k2=(area2*el2)/leng2;
 k12=(k1^2*k2)/(2*k1*k2+k1^2);
 m12=rho1*area1*(leng-leng2)+rho2*area2*leng2;
% total mass of piezoelectric active member
% stiffness matrix

 k13= k12*[ c*c c*s c*r;...
           c*s s*s s*r;...
          c*r s*r r*r];

 k=[k13 -k13;-k13 k13];
% lumped mass matrix

 m=m12/3*eye(6);
%
%                       -----------------------------------            -----
%
```

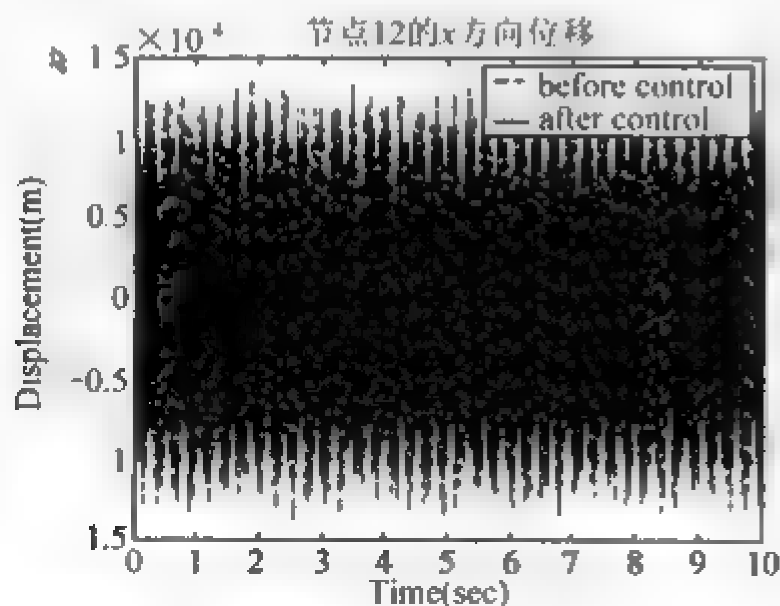
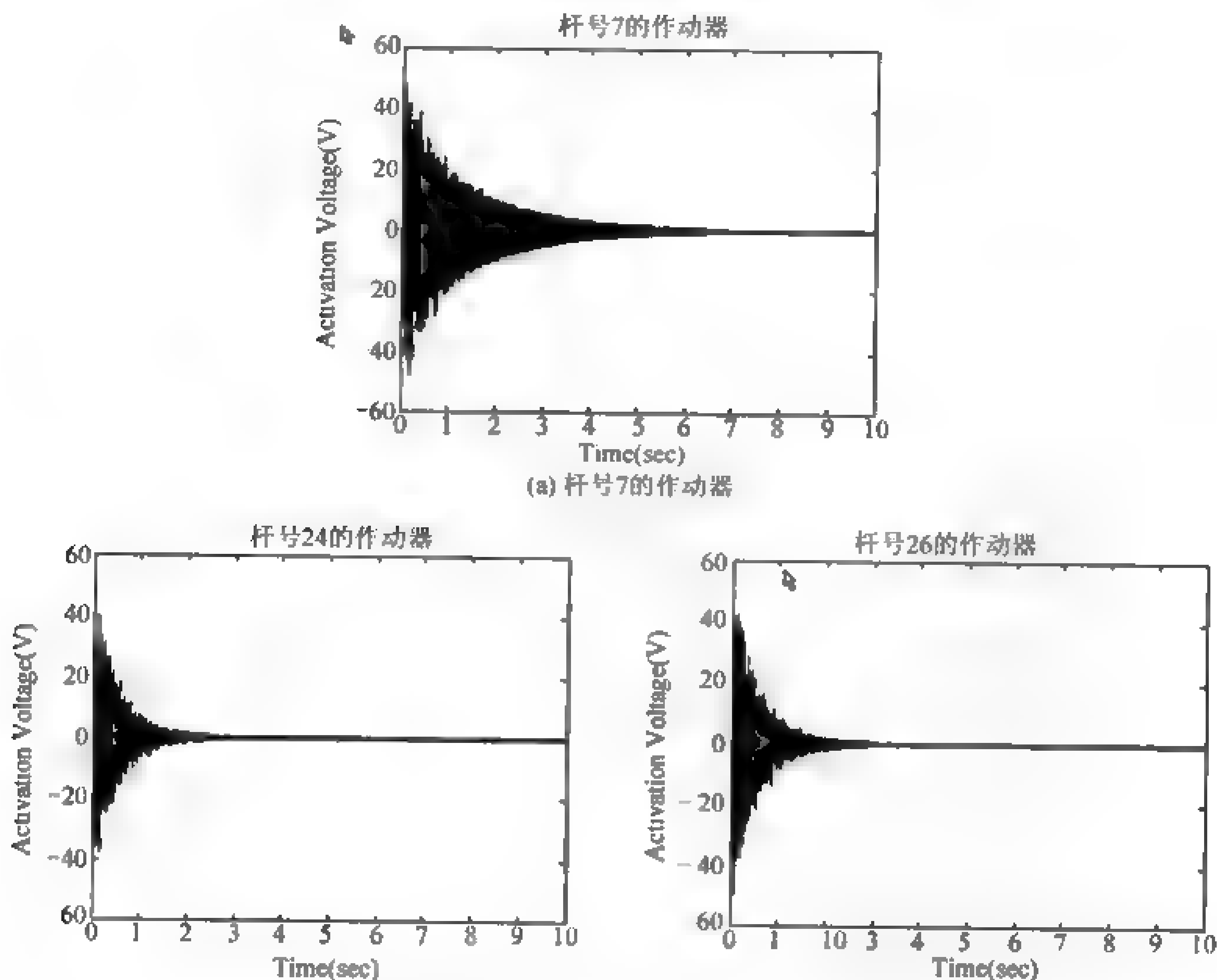


图8 21 控制前后节点12的x方向位移曲线



(a) 杆号7的作动器

图 8 22 作动器输入的电压曲线

8.3 结构边界参数优化设计

多数工程结构是受约束结构，通过特定的边界与其他的物体相连接. 边界结构对结构系统的动力学特性作用明显，且结构边界条件的参数对结构动力学特性的影响也比结构内部的参数更为敏感，因此在一定程度上对结构系统动力学特性的要求可通过边界条件的设计来达到. 由于边界条件的设计变量数一般要比结构内部的参数少，并且有的结构因设计功能和特定的性能约束而使得内部结构不易改变时，利用边界条件的动力学设计来满足结构的动力学特性是一个有效的设计手段.

对于一个结构系统，若以 $\boldsymbol{M}$ 、 $\boldsymbol{K}$ 、 $\boldsymbol{C}$ 分别代表 n 阶正定质量、刚度、阻尼矩阵，并以时间函数 $\boldsymbol{x}(t)$ 、 $\dot{\boldsymbol{x}}(t)$ 、 $\ddot{\boldsymbol{x}}(t)$ 、 $\boldsymbol{p}(t)$ 分别表示试件的位移、速度、加速度和激励力列阵，则系统运动方程为

$$\boldsymbol{M}\ddot{\boldsymbol{x}}(t)+\boldsymbol{C}\dot{\boldsymbol{x}}(t)+\boldsymbol{K}\boldsymbol{x}(t)=\boldsymbol{p}(t) \tag{8-44}$$

两端进行傅氏变换，并记 ω 为傅氏变量，有

$$(\boldsymbol{K}-\omega^2\boldsymbol{M}+j\omega\boldsymbol{C})\boldsymbol{X}(\omega)=\boldsymbol{P}(\omega) \tag{8-45}$$

对于工程结构，阻尼一般为小阻尼，可认为 C 是经典阻尼阵，从而可解得结构系统的固有频率 $\lambda_i=\omega_i^2$，$(i=1,2,\cdots,n)$ 及相应的振型 ϕ_i.

由式(8-45)，可得系统的动位移响应为

$$X(\omega)=\sum_{i=1}^{n}\frac{\phi_i\phi_i^{\mathrm{T}}P}{k_i-\omega^2m_i+j\omega c_i} \tag{8-46}$$

式中，$k_i=\phi_i^{\mathrm{T}}K\phi_i$; $m_i=\phi_i^{\mathrm{T}}M\phi_i$; $c_i=\phi_i^{\mathrm{T}}C\phi_i$.

动位移响应 $X(\omega)$ 是反映结构振动的一个表征量. 多数动响应都可用动位移响应来表示，如 $\dot{X}=j\omega X$、$\ddot{X}=-\omega^2X$、动应力 $\sum=DX$（D 为应力-位移矩阵），因而可用式(8-46)来表征结构的动力特性.

式(8-46)表明，载荷相同时，欲使设计边界条件结构与要求动力学特性的目标结构(称为原结构)或设计边界条件结构(称为设计结构)在有限元模型相近的条件下得到相同的位移幅值，要求两者各阶固有频率及相应振型相同.

在动力学优化设计时既要满足若干阶频率一致，又要使对应的各阶振型相近，是十分困难的. 因此，从工程实际应用出发，结构动力学边界的优化设计采用使前 N 阶固有频率满足设计指标，低阶振型接近的优化策略，使设计结构的动响应与原结构的动响应接近一致.

按照上述的进行结构动力学边界设计的优化策略，可建立如下的优化设计方法.

对于一结构系统，其特征值方程为

$$K\Phi=M\Phi\Lambda \tag{8-47}$$

且满足下列条件

$$\Phi^{\mathrm{T}}M\Phi=I\,,\quad \Phi^{\mathrm{T}}K\Phi=\Lambda \tag{8-48}$$

式中：K 是结构的刚度矩阵；M 是结构的质量矩阵；$\Phi=\begin{bmatrix}\phi_1 & \phi_2 & \cdots & \phi_n\end{bmatrix}$ 是振型矩阵；$\Lambda=\mathrm{diag}\begin{pmatrix}\omega_1^2 & \omega_2^2 & \cdots & \omega_n^2\end{pmatrix}$ 是频率矩阵；I 表示单位矩阵.

对于结构边界动力学设计问题，且只考虑在低阶范围内的动态特性指标，有

$$(K_A^0+K_B^0)\Phi^0=(M_A^0+M_B^0)\Phi^0\Lambda^0 \tag{8-49}$$

$$\Phi_r^{0\mathrm{T}}(M_A^0+M_B^0)\Phi_r^0=I_r\,,\quad \Phi_r^{0\mathrm{T}}(K_A^0+K_B^0)\Phi_r^0=\Lambda_r^0 \tag{8-50}$$

式中：M_B^0、K_B^0 是边界条件的质量和刚度矩阵；M_A^0、K_A^0 是除边界条件外的内部结构的质量和刚度矩阵；Λ^0、Φ^0 是设计结构的固有频率和振型矩阵，$\Phi_r^0=\begin{bmatrix}\phi_1^0 & \phi_2^0 & \cdots & \phi_r^0\end{bmatrix}$，$\Lambda_r^0=\mathrm{diag}\begin{pmatrix}\omega_1^{02} & \omega_2^{02} & \cdots & \omega_r^{02}\end{pmatrix}$，$r\leqslant n$.

边界条件的质量和刚度矩阵则由构成边界结构的结构参数 $b=[b_1,b_2,\cdots,b_s]^{\mathrm{T}}$ 决定，即边界条件的质量和刚度矩阵是边界结构参数的函数，可表示为 $M_B^0(b)$、$K_B^0(b)$. 优化设计通过逐步修改设计参数 b，来改变边界条件的质量和刚度矩阵 $M_B^0(b)$、$K_B^0(b)$，使得所设计边界条件的结构的动态特性指标 Λ_r^0、Φ_r^0 趋于要求的 Λ_r、Φ_r.

以固有频率接近要求频率为约束条件，用范数表示的低阶振型差最小为优化准则，该

优化问题可表示为

$$\text{find} \quad \boldsymbol{b}$$

$$\min \sum_{j} \left\{ \sum_{i}^{n} [(\phi_j(i) - \phi_j^*(i)]^2 \right\}^{\frac{1}{2}}, \quad j = 1,2,\cdots,m \tag{8-51}$$

$$\text{s.t.} \quad g_r(\boldsymbol{b}) = \left| f_r - f_r^* \right| \leqslant \eta f_r^* \ (r = 1,2,\cdots,N)$$

$$\boldsymbol{b}_l \leqslant \boldsymbol{b} \leqslant \boldsymbol{b}_u$$

式中：ϕ_j^*、f_r^* 为原结构要求的振型、固有频率；ϕ_j、f_r 为设计边界结构的振型、固有频率，且 $m \leqslant N$；m、N 分别为要求的振型和频率的阶数；η 为误差限系数；$\boldsymbol{b}_l$、$\boldsymbol{b}_u$ 分别为边界条件设计变量的上、下限.

上述优化问题为非线性约束优化问题，可用序列二次规划法 SQP(Sequential Quadratic Programming)求解.

图 8.23 为一简化的飞机机翼模型，用空间刚架单元模拟的该结构有限元模型如图 8.24 所示. 该模型共有 54 个节点，70 个梁单元，6 个边界支撑. 其中，49～54 为边界上的固定节点，65～70 为边界单元. 机翼内部梁截面尺寸：宽度 14mm，沿 z 向厚度 4mm；边界支撑梁的截面尺寸：宽度 14.8mm，沿 z 向厚度 4mm. 梁材料的参数为：合金钢材料：弹性模量 $E = 2.1\times10^{11}\text{N/m}^2$，泊松比 $\nu = 0.26$，质量密度 $\rho = 7.8\times10^3\text{kg/m}^3$.

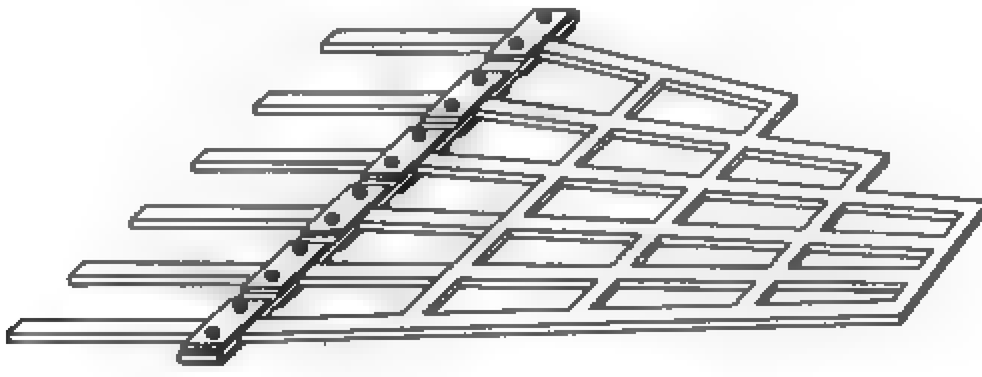


图 8.23 飞机机翼结构

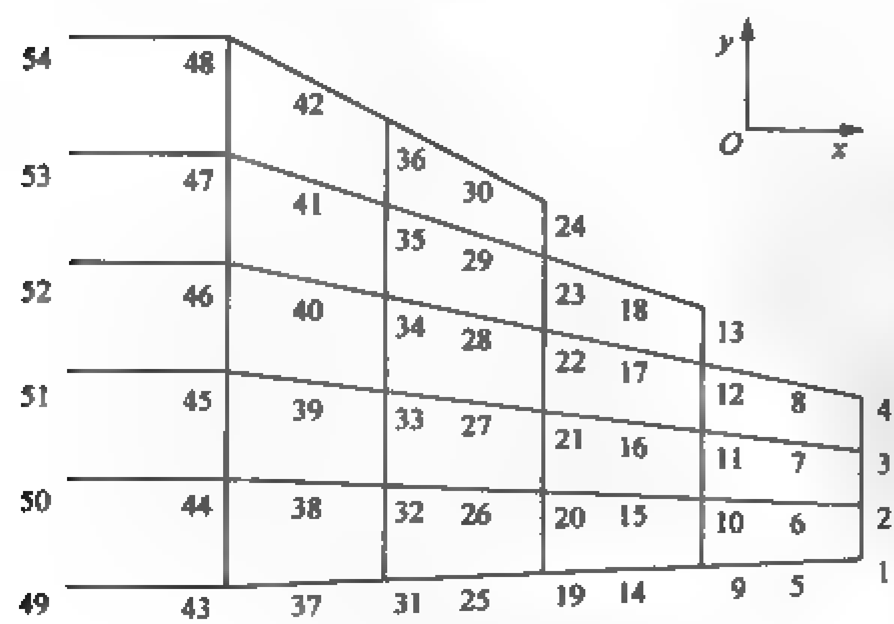


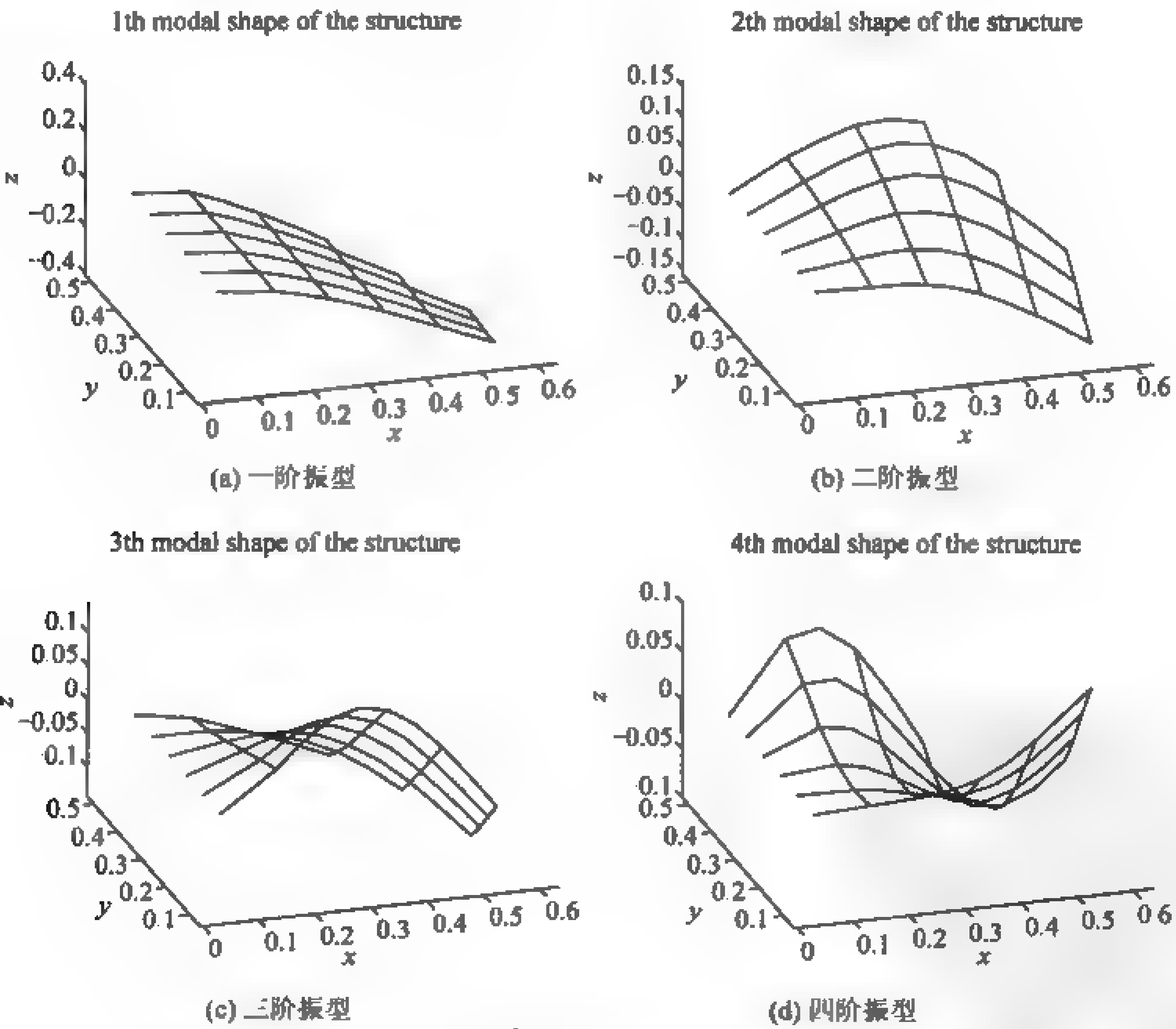
图 8.24 机翼刚架模型

用有限元模型进行计算，得到结构的低阶固有频率和振型见表 8.4 和如图 8.25 所示.

表 8.4　模拟机翼模型固有频率

阶数	1	2	3	4	5
频率(Hz)	12.6779	58.6636	72.7598	135.792	165.761

现用 4 个边界矩形梁支撑机翼，对原结构的边界进行模拟，如图 8.26 所示. 用动态优化设计方法设计梁的截面尺寸，使得结构的前 5 阶固有频率与原结构一致，振型接近. 采用前述的优化策略，以支撑梁的截面参数作为设计变量 $x=[b_i \quad h_i]^{\mathrm{T}}\ (i=1,2,\cdots,s)$，固有频率之差作为约束条件 $g_j(x)=\left|f_j-f_j^*\right|\leqslant \eta f_j^*\ (j=1,2,\cdots,5)$，设计支撑梁结构的一阶、二阶振型与原结构相应振型的范数作为目标函数 $\Delta\varphi=\sum_j\left[\sum_i(\varphi_j(i)-\varphi_j^*(i))^2\right]^{\frac{1}{2}}\ (j=1,2)$，设计的结构边界参数.



(a) 一阶振型　(b) 二阶振型

(c) 三阶振型　(d) 四阶振型

图 8.25　机翼的低阶振型

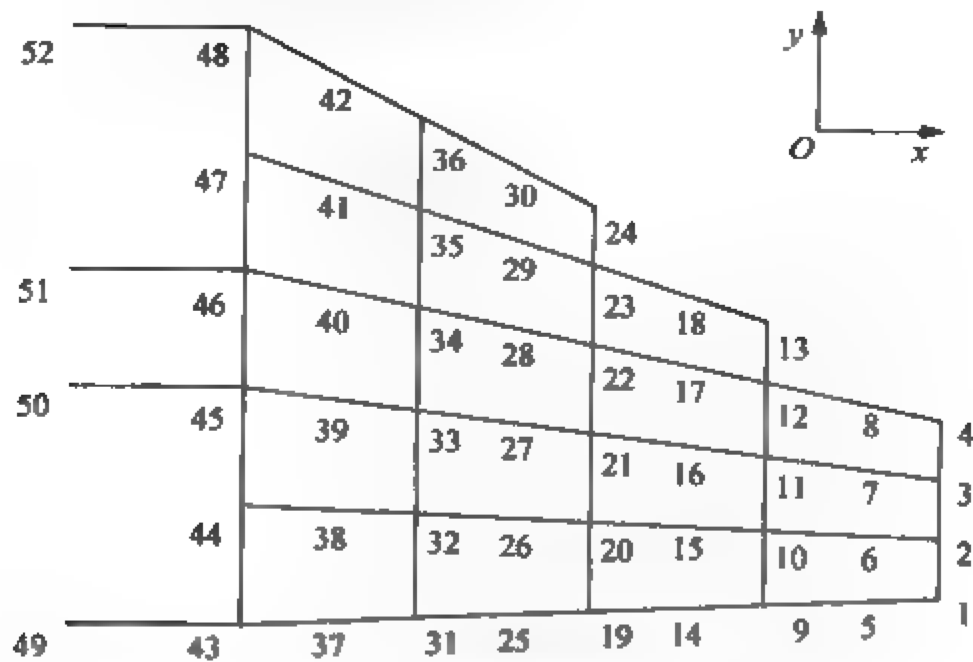


图 8.26 设计边界机翼模型

优化设计的结构振型与原结构振型的一致性可用模态置信准则 MAC(Modal Assurance Criterion)来检验. 对于第 j 阶模态，MAC 的表达式为

$$\mathrm{MAC}_j=\frac{\left|\varphi_j^{*\mathrm{T}}\varphi_j\right|^2}{(\varphi_j^{*\mathrm{T}}\varphi_j^*)(\varphi_j^{\mathrm{T}}\varphi_j)} \tag{8-52}$$

式中，φ_j^* 为原结构的低阶振型，φ_j 为设计边界结构的低阶振型. MAC 的值越接近于 1，振型的符合程度越好.

优化工具用 MATLAB 优化工具箱的函数 fmincon，算法为序列二次规划方法 SQP(Sequential Quadratic Programming). 设计的两组 4 杆支撑机翼分别称为设计结构 1 和设计结构 2，参数见表 8.5.

表 8.5 设计边界结构参数

设计边界结构	结构参数					
设计结构 1	b	0.0148	0.0148	0.0148	0.0148	
	h	0.0044	0.0051	0.0040	0.0046	
	频率	12.506	58.186	72.725	134.13	163.95
	误差	1.36%	0.81%	0.048%	1.22%	1.09%
	$\Delta\varphi$	0.1961				
设计结构 2	b	0.0148	0.0148	0.0148	0.0148	
	h	0.0046	0.0047	0.0048	0.0045	
	频率	12.746	58.915	73.327	135.08	164.31
	误差	0.54%	0.43%	0.78%	0.52%	0.88%
	$\Delta\varphi$	0.1451				

为了验证设计边界结构参数的有效性，对原结构和设计边界支撑的结构进行模态试

验，测试的模态参数见表 8.6.

表 8.6 机翼结构的模态数据

阶数		1	2	3	4	5
原结构	频率	13.1054	59.0111	74.6960	136 0818	167.3189
设计边界结构 1	频率	12.951	58.45	74.926	135.54	169.74
	误差	1.18 %	0.95 %	0.31 %	0.40 %	1.45 %
	MAC	0 985 16	0.981 29	0.983 02	0.829 39	0.973 76
设计边界结构 2	频率	12.94 09	58.2615	75.4238	135.5300	169.4663
	误差	1 26 %	1.27 %	0.97 %	0.41 %	1 28 %
	MAC	0.986 22	0.978 07	0.935 41	0.595 21	0.882 03

由实验结果可知，设计的四杆边界支撑结构的低阶固有频率与原结构基本一致，振型也较符合，对原结构边界有较好的动力学模拟，说明在工程上通过边界结构的动力学模拟来满足特定的动力学特性要求的方法是可行的.

该算例的 matlab 程序如下：

主控程序为 BeamProbB21.

主控程序调用 3 个函数，分别为：

- BeamCom0——计算的目标结构(要求的固有频率和振型由此函数得到).
- BobjFun1——优化的目标函数.
- ConFun1——优化计算的约束条件(此处为不等式约束).
- BeamCom1——优化迭代过程中的有限元计算.

其他函数为有限元计算所用的函数.

```
%--------------------------------------------------------------------
% BeamProbB21  (minimize the difference of lower order modal vectors)
%   BeamProb1: (4 supports)
%   To find the sizes of the structure using optimal algorithm
%
% Variable descriptions
%   Inout:
%         x0 == intial value of the design variables
%   Optput:
%         x == desin varialbes at the optimal solution
%         fval == value of objective function
%--------------------------------------------------------------------
%  (0) input control data
%-----------------------------------
clear; clc;
num_f=5;                        % number of natural frequency
ith_m=2;                         % 1-ith order of modal vectors
kg=0;                           % option for graphic display of modal shape
```

```
global freqt wst Vit
                % freqt -- natrural requency of oririnal structure system
                % wst -- total weight of boundary elements
                % Vit == modal vector of oririnal structure system
%   ---------   ---------   -------   --------   --------  --------
%  (1) calculate the dynamic properties of original structure
%-   --------  ---- -----  --- --------  -------  --------  ------  ---
x0=[0.0148; 0.0148; 0.0148; 0.0148; 0.0148; 0.0148;
   0.0040; 0.0040; 0.0040; 0.0040; 0.0040; 0.0040];
[freqt,wst,Vit]=BeamCom0(x0,num_f,ith_m,kg);   % FEM calculation
%----  -------  --------   ------  ------
%  (2) input variable data
%----------- --------   -------  -------  -
                               % intial value of number of design variables
h0=0.0148; b0=0.005;
x0=[h0; h0; h0; h0;
   b0; b0; b0; b0];   % set the lower and upper bounds of the variables
%hl=0.004; hu=0.020;
%bl=0.004; bu=0.020;
hl=0.0148; hu=0.0148;
bl=0.0040; bu=0.0100;
vlb=[hl; hl; hl; hl;
    bl; bl; bl; bl];
vub=[hu; hu; hu; hu;
    bu; bu; bu; bu];
%----  -------  --------------------------------------  ------  -----
%  (3) process the optimum numerical operation
%--------------   --------  -------  ---------------------------------
% call the fmincon function to get optimal solution
tic
options = optimset('LargeScale','off');
options = optimset('MaxSQPIter', 1000);
options = optimset('MaxFunEvals',2000);
[x,fval]-fmincon(@BobjFun1,x0,[],[],[],[],vlb,vub,@ConFun1,options)
[c, ceq, freq]=ConFun1(x)
te-toc
%--------------------   -------  ------  ---------------------------
%    The end
% --  ---------  -------     -------  ------  --------------------


function [f]-BobjFun1(x)
%-   --- --   -------  ------  -----   ------ -----    -----  --
% Purpose:
%    Calculate the objective function which is the norm of difference of
%    modal vectors
% Synopsis:
```

```
%    [f]=BobjFun1(x)
% Variable Description:
%    x = design variables
%    f = objective function
%-----------------------------------------------------------------------
% (1) prescribe frequency and ith order modal verctor
%-----------------------------------------------------------------------
num_f=5;                    % number of natural frequency
ith_m=2;                    % 1-ith order of modal vectors
kg=0;                       % option for graphic display of modal shape
global Vit
%-----------------------------------------------------------------------
% (2) calculte ith order modal verctor of two structure
%-----------------------------------------------------------------------
[freq,ws,Vi]=BeamCom1(x,num_f,ith_m,kg);          % FEM calculation
                       % freq == natrural requency of structure system
                       % ws == total weight of boundary elements
                       % Vi == modal vectors
%-----------------------------------------------------------------------
% (3) calculate the norm of difference between two modal vectors
%-----------------------------------------------------------------------
f=sum((Vi-Vit).*(Vi-Vit));
f=sum(f);
f=sqrt(f);
%-----------------------------------------------------------------------
%    The end
%-----------------------------------------------------------------------


function [c, ceq, freq] = ConFun1(x)
%-----------------------------------------------------------------------
% Purpose:
%    To computes the nonlinear inequality constraints c(x)<= 0 and the
%    nonlinear equality constraints ceq(x) = 0. This function accepts
%    a vector x and returns two vectors c and ceq. The vector c contains
%    the nonlinear inequalities evaluated at x, and ceq contains the
%    nonlinear equalities evaluated at x.
% Synopsis:
%    [c, ceq]=ConFun1(x)
% Variable Description:
%   Input:
%        x == design variables
%   Optput:
%        c == inequality constraint conditions
%        ceq == equality constraint conditions                    %
%-----------------------------------------------------------------------
% (0) input control data
```

```
%     ---------       --------     -----
num_f=5;                     % number of natural frequency
ith_m=2;                      % 1-ith order of modal vectors
kg=0;                        % option for graphic display of modal shape
global freqt
%  -------      -- -----   --  ---  ------------ ------ --- --------
% (1) calculte the frequency and 1-ith order modal verctors
%-----     -------  --------------  -- -------  ------  -----
[freq,ws,Vi]=BeamCom1(x,num_f,ith_m,kg);                % FEM calculation
                        % freq == natrural requency of structure system
                        % ws == total weight of boundary elements
                        % Vi == modal vectors
%-----   -- - ------- --------- --------  -------  -------  ------ -
% (2) check the boundary conditions
%---  -------  --------  ---------------------------------  ------- -
                                   % Nonlinear inequality constraints
for i=1:num_f
 c(i)=abs(freq(i)-freqt(i))-0.01*freqt(i);
end
                                   % No nonlinear equality constraints
ceq=[];
% ----------------------------------------------------  -------  ---
%    The end
% -----    --------    ------------------------------------------ ---
```

8.4 结构故障诊断

损伤检测、预警及适时维修制度的建立，有助于在一定程度上消除隐患及避免灾难性事故的发生. 作为建立损伤检测、预警及适时维修制度的核心技术问题，如何对结构体系中已经出现的损伤进行有效的识别、定位和量化标定，早在20世纪70年代就已经开始进行研究，结构损伤识别技术近年来得到了巨大发展. 结构损伤检测可分为两类：一类是利用染色渗透、射线、光干涉、超声波和电磁学监测等技术对结构的某些局部进行定期检查. 染色渗透技术是对结构的局部表面进行涂层，涂料则渗透到裂缝里，观察表面就可发现表面的裂纹；射线、射线探损技术是利用构件的射线、射线照片进行损伤识别；超声波技术是向构件发射高频声波并测量折射情况，从而识别出复合材料的分层或其他损伤；电磁学检测技术是利用涡流和磁场的原理进行结构损伤的识别. 这类技术在建筑、航天和船舶等领域有着广泛的应用，但仍有很多缺点：一是对于一些不可见、不开敞的部件难以监测；二是对于一些大型结构特别是比较复杂的大型结构检测其损伤是不可能的；三是这类技术要求监测人员必须到现场才能检测. 可见这类技术仅适用于小型结构的检测. 对于一些不可见、不开敞的部位，该类技术不仅无法实施，甚至要求结构的一些功能停止使用或停止工作. 例如，要求在民用客机停飞、发电机组停止运转的情况下进行测试，是非常不经济

的，甚至无法做到. 于是出现了另外一种局部损伤识别技术，这种技术把传感器(如光纤传感器或压电传感器)固定在一些重要部件中，从而对这些部件进行远距离检测. 这一类技术因为具有可以直接确定构件的裂纹及其位置的优点，所以在公路、桥梁和建筑等许多领域得到应用. 在这一类技术中基于动力特性的损伤识别技术受到了广泛的重视.

目前已经提出的结构动力学损伤振动检测方法非常多，而且各类新的方法还在不断被提出. 即使是对这些方法都仅做简略介绍，在本书有限的篇幅内也是不可能的. 而这些方法都各有其优点，很难选出其中某种具有代表性的方法. 出于这种原因，首先简略介绍一些结构动力学损伤检测的思想，然后再介绍一种比较新的方法.

结构动力学损伤检测的过程大概可以分为 3 步：

(1) 选择一种或多种结构动力学指标进行测量. 包括结构频率、模态、频响函数和响应信号在内的各种动力学损伤指标在各种方法中都被采用过. 目前，一般认为仅仅依靠频率数据不适合完成损伤检测，其他各种动力学指标都有自己的优点，都得到广泛的应用.

(2) 构建结构损伤指标. 从直接使用某种动力学指标作为损伤指标到使用小波分析等各种信号分析方法，利用结构动力学特征构建结构损伤指标的方法也是多种多样.

(3) 通过结构损伤指标识别出结构损伤状态. 由于结构的损伤在结构动力学特征上的反映不会是线性的，加上在构建结构损伤指标时可能引入的非线性因素，在很多方法中，结构损伤指标和结构损伤状态之间的关系是非常复杂的. 很难甚至不能给出一个确定的表达式或者算法来由结构损伤指标得到结构损伤状态信息. 因此在这类方法中，诸如神经网络和遗传算法的模式识别方法得到了广泛应用. 例如，如果证明了某种结构损伤指标对结构中的损伤是敏感的，且是单值的(即不会出现两种不同的结构损伤状态有同样的结构损伤指标)，此时，如果在推导出这种结构损伤指标过程中，并没有给出由结构损伤指标得出识别损伤状态的关系式或者方法，那么可以选择出大量的结构损伤状态，并计算出在这些状态下的结构损伤指标. 然后将这些已知的结构损伤指标和结构损伤状态配对作为训练神经网络的样本，只要样本数目足够，网络规模合适，经过训练后，该神经网络就可以识别此结构中的损伤.

但是也有一些方法，在给出结构损伤指标构建方法的同时，也就指出了结构损伤指标和结构损伤状态间的关系.

下面介绍一种由刘济科和杨秋伟最近提出的一种基于残余力向量的损伤识别方法(A new structural damage identification method，Journal of Sound and Vibration 297，2006：694–703).

8.4.1　基于残余力向量的损伤识别方法

一般认为结构损伤将不会改变结构质量，这样，自由度为 n 的含损伤结构的有限元模型的特征值问题方程为

$$\left(\boldsymbol{K}_d-\lambda_{dj}\boldsymbol{M}\right)\boldsymbol{\phi}_{dj}=0 \tag{8-53}$$

式中：$\boldsymbol{K}_d$、λ_{dj} 和 $\boldsymbol{\phi}_{dj}$ 是含损伤结构的整体刚度矩阵、第 j 阶特征值和第 j 阶特征向量；$\boldsymbol{M}$ 是该结构有限元模型的整体质量矩阵. $\boldsymbol{K}_d$ 可以表示为

$$\boldsymbol{K}_d=\boldsymbol{K}_u-\Delta\boldsymbol{K} \tag{8-54}$$

式中，$\Delta\boldsymbol{K}$ 是由损伤引起的结构整体刚度矩阵变化量.

将式(8-54)带入式(8-53)得到

$$\left(\boldsymbol{K}_u-\lambda_{dj}\boldsymbol{M}\right)\boldsymbol{\phi}_{dj}=\Delta\boldsymbol{K}\boldsymbol{\phi}_{dj} \tag{8-55}$$

记 $\boldsymbol{b}_j=\left(\boldsymbol{K}_u-\lambda_{dj}\boldsymbol{M}\right)\boldsymbol{\phi}_{dj}$，则式(8-55)可以写为

$$\Delta\boldsymbol{K}\boldsymbol{\phi}_{dj}=\boldsymbol{b}_j \tag{8-56}$$

如果共测量了 m 个结构模态，对于每一个测量的模态，可以得到如式(8-56)的一个表达式，将各表达式合并起来写成矩阵形式为

$$\Delta\boldsymbol{K}\boldsymbol{\Phi}=\boldsymbol{B} \tag{8-57}$$

式中：$\boldsymbol{\Phi}=\begin{bmatrix}\boldsymbol{\phi}_{d1} & \boldsymbol{\phi}_{d2} & \cdots & \boldsymbol{\phi}_{dm}\end{bmatrix}$；$\boldsymbol{B}=\begin{bmatrix}\boldsymbol{b}_1 & \boldsymbol{b}_2 & \cdots & \boldsymbol{b}_m\end{bmatrix}$.

结构整体刚度矩阵变化量 $\Delta\boldsymbol{K}$ 可以表示为

$$\Delta\boldsymbol{K}=\boldsymbol{A}\Delta\boldsymbol{P}\boldsymbol{A}^{\mathrm{T}} \tag{8-58}$$

式中，$\boldsymbol{A}$ 是刚度连通矩阵(stiffness connectivity matrix)，$\Delta\boldsymbol{P}$ 为单元损伤参数矩阵.

$$\boldsymbol{A}=\begin{bmatrix}\boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_n\end{bmatrix} \tag{8-59}$$

$$\Delta\boldsymbol{P}=\begin{bmatrix}\alpha_1 & & & \\ & \alpha_2 & & \\ & & \ddots & \\ & & & \alpha_n\end{bmatrix},\alpha_i\in[0,1] \tag{8-60}$$

其中含有 n 个元素的向量 $\boldsymbol{a}_i$ 是第 i 个单元刚度连通向量，n 是结构有限元模型中的单元总数，α_i 第 i 个单元损伤参数. 如果单元未损伤，则相应单元 α_i 的值为 0；如果该单元完全损伤，则相应值为 1. 无论有无损伤，$\boldsymbol{A}$ 是不变的(为不影响此处推导过程的连续性，计算 $\boldsymbol{A}$ 矩阵的算法将在本节稍后介绍). 根据 $\Delta\boldsymbol{P}$ 的定义，可见 $\Delta\boldsymbol{P}$ 中的非零元素数目即结构中的损伤数目，同时也等于 $\Delta\boldsymbol{P}$ 的秩.

将式(8-58)带入式(8-57)得到

$$\boldsymbol{A}\Delta\boldsymbol{P}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{\Phi}=\boldsymbol{B} \tag{8-61}$$

在上式两边左乘 $\boldsymbol{\Phi}^{\mathrm{T}}$ 得到

$$\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{A}\Delta\boldsymbol{P}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{\Phi}=\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{B} \tag{8-62}$$

令

$$\boldsymbol{C}=\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{A} \tag{8-63}$$

$$\boldsymbol{D}=\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{B} \tag{8-64}$$

将式(8-63)和式(8-64)带入式(8-62)中得到

$$\boldsymbol{C}\Delta\boldsymbol{P}\boldsymbol{C}^{\mathrm{T}}=\boldsymbol{D} \tag{8-65}$$

其中 m 阶的方阵 $\boldsymbol{D}$ 称为损伤矩阵. 其特征向量分解为

$$\boldsymbol{D}=\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\mathrm{T}} \tag{8-66}$$

其中

$$\boldsymbol{U}=\begin{bmatrix}\boldsymbol{u}_1 & \boldsymbol{u}_2 & \cdots & \boldsymbol{u}_m\end{bmatrix} \tag{8-67}$$

$$\boldsymbol{\Lambda}=\mathrm{diag}\left(\sigma_1 \quad \sigma_2 \quad \cdots \quad \sigma_n\right) \tag{8-68}$$

将式(8-66)带入式(8-65)中得到

$$C\Delta PC^{\mathrm{T}} = U\Lambda U^{\mathrm{T}} \tag{8-69}$$

从式(8-69)中可以看出，ΔP 的秩应当和 Λ 的秩相等. 由于 Λ 是对角矩阵，所以其对角元素中的非零元素个数即为 Λ 的秩(此结论证明详见文献[25])，因此可以通过 Λ 的秩来得知结构中损伤单元的个数. 在有噪声污染测量数据的情况下，也可以通过 Λ 中对角线上明显较大的元素的数量来确定结构的损伤单元数目.

式(8-69)等同于

$$\Lambda = U^{\mathrm{T}} C\Delta PC^{\mathrm{T}} U \tag{8-70}$$

令

$$E = U^{\mathrm{T}} C = \begin{bmatrix} e_1 & e_2 & \cdots & e_n \end{bmatrix} \tag{8-71}$$

式中，$m \times N$ 阶的矩阵 E 称为损伤定位矩阵. 该矩阵的每一列对应于结构中的一个单元，称为损伤定位向量，当单元损伤后，相关列中与 Λ 对角线上零元素对应的元素将全部为零(证明详见文献[25]). 因此可以通过矩阵 E 来定位结构中的损伤.

在确定了损伤所在单元后，可以进一步获知单元损伤程度. 假设由以上过程已知结构中有 q 个损伤单元，其对应的损伤定位向量分别为：$e_1, e_2, \cdots, e_n$，这些向量可以组装为 $m \times q$ 阶的矩阵 S

$$S = \begin{bmatrix} e_1 & e_2 & \cdots & e_n \end{bmatrix} \tag{8-72}$$

在 S 中将有 $m-q$ 行零元素，式(8-70)可以缩减成

$$\Lambda = S\Delta P^{*} S^{\mathrm{T}} \tag{8-73}$$

其中

$$\Delta P = \begin{bmatrix} \alpha_1 & & & \\ & \alpha_2 & & \\ & & \ddots & \\ & & & \alpha_q \end{bmatrix} \tag{8-74}$$

将 S 中元素全部为零的行的移去，$m \times q$ 的矩阵 S 将缩减成 $q \times q$ 的矩阵 S^{*}. 相应地，也将 Λ 中对角线上元素为零的行列移去，m 阶方阵 Λ 将缩减成 q 阶方阵 Λ^{*}. 这样，式(8-73)变成

$$\Lambda^{*} = S^{*} \Delta P^{*} S^{*\mathrm{T}} \tag{8-75}$$

这样，可以获得损伤程度

$$\Delta P^{*} = S^{*-1} \Lambda^{*} \left(S^{*\mathrm{T}}\right)^{-1} \tag{8-76}$$

刚度连通矩阵 A 可以通过以下方法计算.

因为 ΔP 是对角矩阵，所以根据式(8-58)，ΔK 中元素的值实际可以写成：

$$\Delta K_{ij} = \left\{ A_{i1} A_{j1} \quad A_{i2} A_{j2} \quad \cdots \quad A_{in} A_{jn} \right\} \begin{Bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{Bmatrix} \tag{8-77}$$

根据整体刚度矩阵 $\boldsymbol{K}$ 中元素对各单元刚度变化的敏感关系，可以得到 $\Delta\boldsymbol{K}$ 中元素值的另一个表达式

$$\Delta K_{ij}=\frac{\partial K_{ij}}{\partial \alpha_{\beta}} \tag{8-78}$$

综合以上两式可以得到

$$A_{i\beta}A_{j\beta}=\frac{\partial K_{ij}}{\partial \alpha_{\beta}} \tag{8-79}$$

在一般的有限元单元中，单元刚度矩阵的值与单元刚度参数之间是线性关系，所以有

$$\frac{\partial K_{ij}}{\partial \alpha_{\beta}}=\frac{\Delta K_{ij}}{\Delta \alpha_{\beta}} \tag{8-80}$$

结合式(8-79)和式(8-80)即可计算出刚度连通矩阵 A.

8.4.2 数值验证

下面使用该方法对第 3 章中计算过的 5 节点和 7 杆件的桁架模型进行损伤检测. 该桁架各杆件的截面面积统一为 A=1e-4m^3，弹性模量为 E=2.1e11Pa，质量密度为 $\rho=7300\text{kg/m}^3$. 结构在 1 号节点处有水平和垂直方向上的位移约束，在 3 号节点上有垂直方向上的位移约束，如图 8.27 所示.

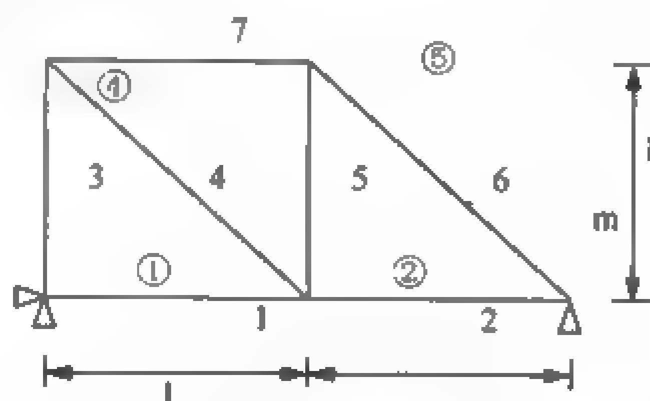


图 8.27 桁架结构

假设在第 3 根杆件上发生损伤，令杆件刚度降低 15%. 下面通过前面介绍的方法来检测该损伤.

首先使用第 3 章中的程序获得该桁架结构完好时的整体刚度矩阵和质量矩阵，再对程序进行调整(程序见后)，计算出损伤后的模态和频率等数据. 在这里通过数值仿真来演示该方法，所以通过有限元模型计算出损伤后的数据；实际应用中，损伤后的数据应采用实验测量结果.

采用损伤后结构的前 3 阶模态数据，根据式(8-55)～式(8-57)求出矩阵 $\boldsymbol{B}$:

```
B = 1.0e+006 *

  [ 0.0000     0.0000    -0.0000
```

```
 -0.0000    0.0000   -0.0000
  0.0000    0.0000   -0.0000
 -0.0000   -0.0000    0.0000
  0.5513    1.0454   -2.2701
  0.0000    0.0000    0.0000
  0.0000    0         0.0000]
```

根据式(8-64)求出矩阵 $\boldsymbol{D}$:

```
D =  1.0e+006 *

  [0.0965  -0.1830   0.3973
  -0.1830   0.3470  -0.7534
   0.3973  -0.7534   1.6359]
```

求矩阵 $\boldsymbol{D}$ 的特征值和特征向量:

[U,Lambda]=eig(D);

(Lambda 即式(8-66)中的 $\boldsymbol{\Lambda}$)

得到:

```
U =
  [-0.2154  -0.8432  -0.0538
    0.4085   0.3804  -0.9119
   -0.8870   0.3800  -0.4069]
Lambda =
  [2.0794        0        0
        0  -0.0000        0
        0        0   0.0000]
```

可以看出在 $\boldsymbol{\Lambda}$ 矩阵的对角线元素上，有一个明显非零元素和两个近似为零的元素(两个近似为零的元素值分别为-1.3533e-008 和 9.4201e-009，可以认为是计算过程中的数值误差)因此可知，在结构中有一处损伤.

下面来进一步确定损伤的位置.

通过式(8-79)和式(8-80)求出矩阵 $\boldsymbol{A}$:

```
A =  1.0e+003 *
   [4.5826   4.5826        0    2.7248    0.0000         0         0
         0        0        0   -2.7248   -4.5826         0         0
         0  -4.5826        0         0         0    2.7248         0
         0        0   0.0000   -2.7248         0         0    4.5826
         0        0  -4.5826    2.7248         0         0         0
         0        0        0         0   -0.0000   -2.7248   -4.5826
         0        0        0         0    4.5826    2.7248        0]
```

再根据式(8-63)求出矩阵 $\boldsymbol{C}$:

```
C =  1.0e+003 *
  [-1.7265   0.8577   0.8021   0.9702  -0.6483   1.0406  -0.0224
   -0.0615   1.0588  -1.5209  -1.7020  -0.0817   1.5348  -1.0601
```

```
    4.0309   -0.5516    3.3024    0.6201    2.0540    3.0960   -1.1669]
```

根据式(8-71)求出矩阵 $\boldsymbol{E}$:

```
E =  1.0e+003 *
  [ -3.2285    0.7370   -3.7232   -1.4542    1.9281   -2.3433    0.6068
     2.9639   -0.5300    0.0000   -1.2298   -0.2649    0.8828   -0.8277
    -1.4912   -0.7872   -0.0000    1.2475    0.9451   -2.7153    1.4427]
```

刚才已经得知在 $\boldsymbol{A}$ 矩阵对角线元素上，第 2 个和第 3 个元素为零. 在 $\boldsymbol{E}$ 矩阵与此相对应的第 2 行和第 3 行中，只有第 3 列上的两个元素为零. 因此可以得知，结构损伤发生在第 3 个单元中.

最后求解单元损伤程度. 由于已知单元中只有一个损伤，所以 $\boldsymbol{A}^*$ 和 $\boldsymbol{S}^*$ 都只有一个元素，式(8-76)的计算为标量计算.

$$\Delta \boldsymbol{P}^* = \boldsymbol{S}^{*-1}\boldsymbol{A}^*\left(\boldsymbol{S}^{*\mathrm{T}}\right)^{-1} = (-3.7232e+003)^{-1} * 2.0794 * (-3.7232e+003)^{-1} = 0.15$$

与实际的刚度损失 15%完全相符.

这种方法也可以完成对结构中多个同时存在损伤的识别，且在模态数据中含有随机干扰的情况下，也能获得良好的识别结果.

用以分析损伤后结构的 MATLAB 程序如下:

```
E=2.1e11;
A=1e-4;
density=7.3e3;
node_number=5;
element_number=7;

nc=[0,0;1,0;2,0;0,1;1,1];
%nc:node_coordinate

en=[1,2;2,3;1,4;2,4;2,5;3,5;4,5];
%en:element_node
ed(1:node_number,1:2)=1;
%ed:element_displacement
constraint=[1,1;1,2;3,2];

for loopi=1:length(constraint);
    ed(constraint(loopi,1),constraint(loopi,2))=0;
end
dof=0;
for loopi=1:node_number
     for loopj=1:2
        if ed(loopi,loopj)~=0
            dof=dof+1;
            ed(loopi,loopj)=dof;
        end
    end
```

```
end

%el:length of link element
ek=E*A*[1 0 -1 0;0 0 0 0; 1 0 1 0;0 0 0 0];
%ek:element_stiffness_matrix
em=(density*A)/2*eye(4);
%em:element_mass_matrix
k(1:dof,1:dof)=0;
%k:structural_stiffness_matrix
m=k;
%m:structural_mass_matrix,same size with k
theta(1:7)=0;
el(1:7) 0;
e2s(1:4)=0;
% e2s: index of transform the element displament number to structural
for loopi=1:element_number
    for zi=1:2
        e2s((zi-1)*2+1)=ed(en(loopi,zi),1);
        e2s((zi-1)*2+2)=ed(en(loopi,zi),2);
    end

    el(loopi)=sqrt((nc(en(loopi,1),1)-nc(en(loopi,2),1))^2
    +(nc(en(loopi,1),2)-nc(en(loopi,2),2))^2);
    theta(loopi)=asin((nc(en(loopi,1),2) nc(en(loopi,2),2))/el(loopi));
    lmd=[cos(theta(loopi)) sin(theta(loopi)); -sin(theta(loopi))
    cos(theta(loopi))];
    t=[lmd zeros(2); zeros(2) lmd];
    dk=t'*ek*t/el(loopi);
    dm=t'*em*t*el(loopi);

    for jx=1:4
        for jy=1:4
            if(e2s(jx)*e2s(jy)~=0)
                if (loopi~=3) % damage incurred in 3th element.
                    k(e2s(jx),e2s(jy))=k(e2s(jx),e2s(jy))+dk(jx,jy);
                else
                    k(e2s(jx),e2s(jy))=k(e2s(jx),e2s(jy))+0.85*dk(jx,jy);
                    % introduce damage which reduce 15% of the elemental stiffness
                end
                m(e2s(jx),e2s(jy))=m(e2s(jx),e2s(jy))+dm(jx,jy);
            end
        end
    end
end

[v,d] = eig(k,m);
```

8.5　转子动力学分析

8.5.1　Newmark-β 数值算法

在对简单转子系统做理论分析时，通常都会建立转子的运动微分方程，若考虑其中一些非线性现象(例如挤压油膜阻尼器、轴上含裂纹、转静件碰磨等)，这个方程就变为非线性运动微分方程. 求解微分方程(特别是非线性微分方程)通常采用数值解法，常见的算法有Runge-Kutta法、Euler后差法、Newmark-β 法等，这些算法各有特点.

MATLAB软件中的数学工具箱采用的算法均为改进的Runge-Kutta法，可以自适应改变步长，但算法本身收敛速度慢，容易发散，并且积分步长对结果的精度和正确性有很大影响；另外，在进行数值仿真时通常需要得到系统的稳态解，这需要进行长时间的仿真(舍去由于不同初值导致的瞬态响应)，而MATLAB软件的Runge-Kutta法在求解时是将每一步的计算结果作为矩阵一行元素加在以前仿真结果之后，长时间的仿真就会产生一个非常大的矩阵，占用很大的内存，导致计算机在进行频繁的数据交换，大大降低了计算速度，这些不足使得它的使用范围受到很大限制. Euler后差法具有特殊、优越的稳定性，精度低可以通过减小步长来弥补，积分步数增加而带来的机时增加可由简单的差分格式得到补偿. 而Newmark-β 法的优点在于当参数选择合适时积分是无条件稳定的，其结果与步长大小无关，并且“Newmark-β 法是几种常用线性加速度法中的最优算法”，就单步误差而言，其截断误差为 $O(\Delta t^3)$，比Euler后差法的截断误差 $O(\Delta t)$ 小，计算精度高. 因此，本节将介绍Newmark-β 法在求解裂纹转子的运动微分方程的应用.

1. Newmark-β 法简介

Newmark-β 法是在计算时假设加速度为介于 $\ddot{u}_t$ 和 $\ddot{u}_{t+\Delta t}$ 之间的某一常向量的一种方法，即

$$\{\ddot{u}\}=\{\ddot{u}_t\}+\delta(\{\ddot{u}_{t+\Delta t}\}-\{\ddot{u}_t\}) \tag{8-81}$$

也可采用

$$\{\ddot{u}\}=\{\ddot{u}_t\}+2\alpha(\{\ddot{u}_{t+\Delta t}\}-\{\ddot{u}_t\}) \tag{8-82}$$

式中，α,δ 为控制参数，它们决定着积分的稳定性及精度，Δt 为积分步长. 当 $\delta>0.5, a>0.25(\delta+0.5)^2$ 时积分是无条件稳定的.

经过积分得到

$$\begin{aligned}\{\dot{u}_{t+\Delta t}\}&=\{\dot{u}_t\}+[(1-\delta)\{\ddot{u}_t\}+\delta\{\ddot{u}_{t+\Delta t}\}]\Delta t\\ \{u_{t+\Delta t}\}&=\{u_t\}+\Delta t\{\dot{u}_t\}+[(0.5-\alpha)\{\ddot{u}_t\}+\alpha\{\ddot{u}_{t+\Delta t}\}]\Delta t^2\end{aligned} \tag{8-83}$$

令

$$a_1=\frac{1}{\alpha\Delta t^2}\qquad a_2=-\frac{1}{\alpha\Delta t}\qquad a_3=-(\frac{1}{2\alpha}-1)$$

$$b_1=\frac{\delta}{\alpha\Delta t}\qquad b_2=(1-\frac{\delta}{\alpha})\qquad b_3=(1-\frac{\delta}{2\alpha})\Delta t$$

则由式(8-82)可求出由 $\{u_{t+\Delta t}\}$、$\{\ddot{u}_t\}$、$\{\dot{u}_t\}$ 及 $\{u_t\}$ 表示的 $\{\ddot{u}_{t+\Delta t}\}$ 和 $\{\dot{u}_{t+\Delta t}\}$

$$\{\ddot{u}_{t+\Delta t}\} = a_1(\{u_{t+\Delta t}\} - \{u_t\}) + a_2\{\dot{u}_t\} + a_3\{\ddot{u}_t\}$$
$$\{\dot{u}_{t+\Delta t}\} = b_1(\{u_{t+\Delta t}\} - \{u_t\}) + b_2\{\dot{u}_t\} + b_3\{\ddot{u}_t\} \tag{8-84}$$

将运动方程写成如下统一形式

$$[M]\{\ddot{u}\} + [C]\{\dot{u}\} + [K]\{u\} = \{g(t)\} + \{f\} \tag{8-85}$$

将式(8-83)代入式(8-85)得

$$\overline{K}\{u_{t+\Delta t}\} = \{\overline{g}\} + \{f_{t+\Delta t}\} \tag{8-86}$$

其中

$$[\overline{K}] \equiv a_1[M]_{t+\Delta t} + [K]_{t+\Delta t} + b_1[C]_{t+\Delta t}$$

$$\{\overline{g}\} \equiv \{g(t+\Delta t)\} + [M]_{t+\Delta t}\{a_1\{u_t\} - a_2\{\dot{u}_t\} - a_3\{\ddot{u}_t\}\} + [C]_{t+\Delta t}\{b_1\{u_t\} - b_2\{\dot{u}_t\} - b_3\{\ddot{u}_t\}\}$$

对于如式(8-84)的微分方程，则可以根据式(8-83)和式(8-85)求出其响应.

2. 程序编制

【例 8.11】 考虑 Jeffcott 转子系统模型，其运动微分方程为

$$m\ddot{x} + 2mD\omega_{nz}\dot{x} + k_z x = me\omega^2\sin(\omega t + \phi)$$
$$m\ddot{y} + 2mD\omega_{nz}\dot{y} + k_z y = me\omega^2\cos(\omega t + \phi) \tag{8-87}$$

对式(8-87)进行无量纲化，左右两端同除以 $m\delta\omega^2$

$$X'' + \frac{2D}{\Omega}X' + \frac{1}{\Omega^2}X = U\sin(\tau + \phi)$$
$$Y'' + \frac{2D}{\Omega}Y' + \frac{1}{\Omega^2}Y = U\cos(\tau + \phi) \tag{8-88}$$

其中，无量纲参数为

$$X = \frac{x}{\delta},\quad Y = \frac{y}{\delta},\quad \Omega = \frac{\omega}{\omega_{nz}},\quad \tau = \omega t,\quad U = \frac{e}{\delta},$$

$\delta = mg/k_z$ 为转子的静位移.

转子模型参数如下：$l = 1\text{m}, m = 100\text{kg}, R = 0.03\text{m}, E = 200\text{GPa}$，重力加速度取 $g = 9.8\ \text{N/m}^2$，转盘半径 $R_d = 0.3\text{m}$，转盘偏心距为 $2\times10^{-4}\text{m}$，阻尼系数 $D = 0.01$. 对式(8-88)进行仿真，得到系统响应如图 8.28 所示(均为稳态响应).

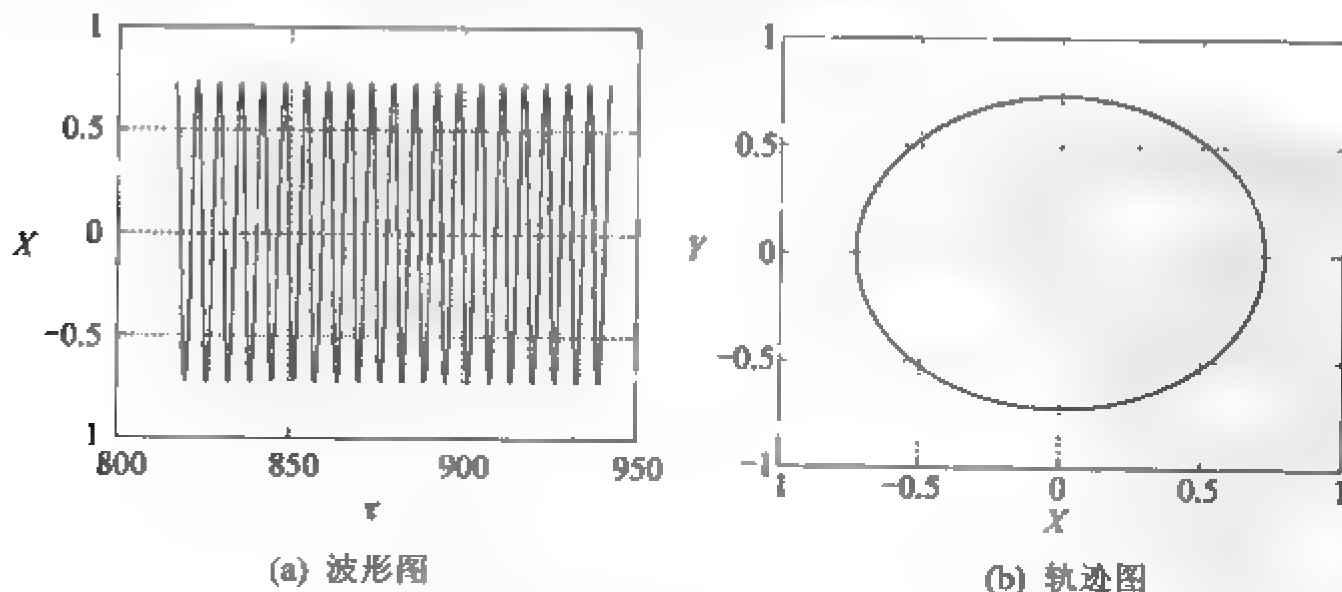


图 8.28　MATLAB 计算环境下 Newmark-β 法仿真结果

MATLAB 程序：

```
E=2*10^11;
L=1;
m1=100;
R=0.03;
e=2*10^(-4);
g1=9.8;
w=150;
dt=pi/500;
t0=0;
I=pi*R^4/4;
kz=48*E*I/L^3;
wn=(kz/m1)^0.5;
q=m1*g1/kz;

U=e/q;
W=w/wn;

r=0.5;
a=0.25;
a1=1/(a*dt^2);
a2=-1/(a*dt);
a3=-(0.5/a-1);
b1=r/(a*dt);
b2=1-r/a;
b3=(1-0.5*r/a)*dt;
m=eye(2);
c=2*[0.01/W 0;0 0.01/W];
k=[1/W^2 0;0 1/W^2];
g=[U*sin(t0);U*cos(t0)];

u0=[0;0];
up0=[0.001;0.001];
upp0=m^(-1)*(g-k*u0-c*up0);

u=u0;
up=up0;
upp=upp0;
fid=fopen('zhuanzi.txt','w');
for t=(t0+dt):dt:300*pi
    g=[U*sin(t);U*cos(t)];
    K=a1*m+b1*c+k;
    G=g+m*(a1*u-a2*up-a3*upp)+c*(b1*u-b2*up-b3*upp);
    u1=K^(-1)*G;
    up1=b1*(u1-u)+b2*up+b3*upp;
    upp1=a1*(u1-u)+a2*up+a3*upp;
```

```
    u=u1;
    up=up1;
    upp=upp1;
    fprintf(fid,'%f    %f     %f\n',u1(1),u1(2),t);
end
fclose(fid);
```

8.5.2 影响系数法进行双面转子动平衡

1. 影响系数法介绍

影响系数法是基于校正配重与测量的机器振动之间的线性关系，即影响系数，来对转子进行平衡的. 对于柔性转子，采用两平面影响系数法，需运行转子3次实现平衡.

如图 8.29 所示，使转子在原配轴承基座上以工作转速或者其他选定的平衡转速转动. 在机器上选定两个测振点 1、2，速度、加速度传感器通常选在左右两个轴承盖上，对于位移传感器，可选在轴颈上或者转子本体上. 光电传感器输出键相脉冲信号，可得到转速以及用以计算相位. 为了消除来自于其他机器的振动干扰和机器本身不平衡以及其他因素的干扰，需要对原始振动信号进行滤波处理，滤掉与转速不同频的信号. 振动信号和所加的试重都是矢量，为了方便地进行复数运算，通常需将一阶滤波后的数字信号及试重矢量进行幅值和相位的分离(MATLAB 可进行复数运算，无需此步). 试重矢量的幅值采用试重矢量矩，即取试重大小与加重半径的乘积，相位取加试重处在相位标记面上对应的相位值.

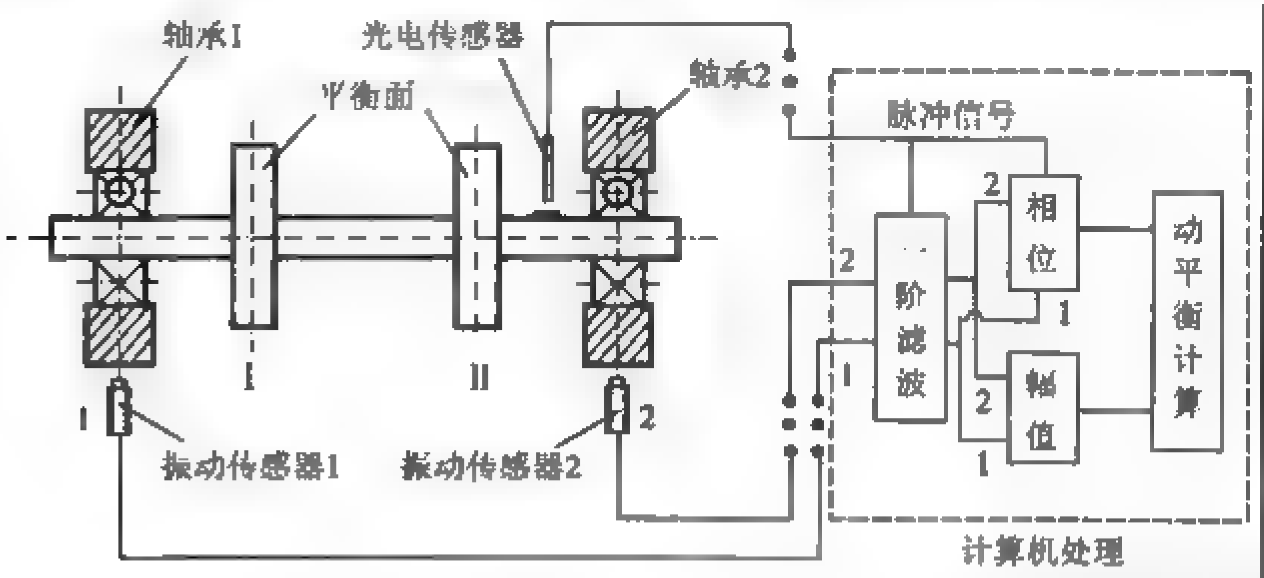


图 8.29 影响系数法对转子现场平衡

用影响系数法平衡刚性转子的实施过程分为以下3个步骤(转子3次运行)：

1) 第0次运行

使转子以转速Ω运行，在测点1和2处测得转子的振动分别为$\dot{X}_{10}$和$\dot{X}_{20}$，作为平衡前的原始值. 此测量信号包含了原始不平衡、支承以及机匣特性的影响.

2) 第1次运行

根据转子结构选取两个校正面I、II. 先在平面I上加一试重$\dot{u}_1$，使转子仍以转速Ω运行. 在测点1和2处测得转子的振动分别为$\dot{X}_{11}$和$\dot{X}_{21}$. 它们既包含了原始不平衡、支承以

及机匣特性的影响，也包含了试重$\vec{u}_1$的影响. 故影响系数为

$$\vec{\alpha}_{11}=\frac{\vec{X}_{11}-\vec{X}_{10}}{\vec{u}_1} \qquad \vec{\alpha}_{21}=\frac{\vec{X}_{21}-\vec{X}_{20}}{\vec{u}_1}$$

3) 第2次运行

取走试重$\vec{u}_1$，在平面Ⅱ上加试重$\vec{u}_2$，再次使转子以速度Ω运行. 在测点 1 和 2 处测得转子的振动分别为$\vec{X}_{12}$和$\vec{X}_{22}$. 试重的影响系数为

$$\vec{\alpha}_{12}=\frac{\vec{X}_{12}-\vec{X}_{10}}{\vec{u}_2} \qquad \vec{\alpha}_{22}=\frac{\vec{X}_{22}-\vec{X}_{20}}{\vec{u}_2}$$

求得影响系数后，列出平衡条件

$$\begin{Bmatrix}\vec{X}_{10}\\ \vec{X}_{20}\end{Bmatrix}+\begin{Bmatrix}\vec{\alpha}_{11} & \vec{\alpha}_{12}\\ \vec{\alpha}_{21} & \vec{\alpha}_{22}\end{Bmatrix}\begin{Bmatrix}\vec{u}_1\\ \vec{u}_2\end{Bmatrix}=C \tag{8-89}$$

可求得校正量$\vec{u}_1$和$\vec{u}_2$

$$\begin{Bmatrix}\vec{u}_1\\ \vec{u}_2\end{Bmatrix}=-\begin{Bmatrix}\vec{\alpha}_{11} & \vec{\alpha}_{12}\\ \vec{\alpha}_{21} & \vec{\alpha}_{22}\end{Bmatrix}^{-1}\begin{Bmatrix}\vec{X}_{10}\\ \vec{X}_{20}\end{Bmatrix} \tag{8-90}$$

用解析法解此矢量方程组，可求得校正块质量及位置.

在转子上安装校正质量，重新启动转子. 如振动已减小到满意程度，则平衡结束. 否则可再一次进行修正平衡. 一般来说，若转子系统无异常，振动的测量和计算是正确的，一两次加重就可以得到满意的结果.

2. 程序编制

【例 8.12】 对 Bently 模拟进行动平衡，其实验表格见表 8.7. 动平衡过程共分以下 7 个步骤：①低转速测量初始弯曲(由轴承内环不光滑、轴初始弯曲、盘面不光滑等导致)；②第 0 次运行，测得转子的原始振动；③第 1 次运行，测得转子在 1 面上加配重的振动；④第 2 次运行，测得转子在 2 面上加配重的振动；⑤计算需加配重大小；⑥在转子上实加配重；⑦加上配重以后第 3 次运行，测得平衡结果. 其测量数据是盘附近轴的振动位移量，因转盘只能在转盘固定角度施加配重，并且所能施加的最大配重为 2g，因此，实际操作时将配重分别加在 3 个角度上，使其合成配重与计算配重接近. 因实验转子加工精度比较高，所以没有对数据进行滤波. 其在水平方向的初始弯曲、原始振动和平衡结果的波形图如图 8.30 所示.

表 8.7 Bently 模拟转子动平衡数据

步骤	试重				一阶振动量			
	1面		2面		1面		2面	
	大小	相位	大小	相位	幅值	相位	幅值	相位
200转(初始弯曲)					0.0436	290.0	0.0390	299.3
第0次运行					0.2186	271.9	0.2057	277.1
第1次运行	1.2	90			0.1175	281.6	0.1158	292.0

续表

步骤	试重				一阶振动量			
	1 面		2 面		1 面		2 面	
	大小	相位	大小	相位	幅值	相位	幅值	相位
第 2 次运行			1.2	90	0.1215	278.4	0.1136	290.3
计算配重	1 面 5.84，角度 48；2 面 5.05，角度 200.4							
实加配重	1 面 22.5 度加 1.6，45 度加 2，67.5 度加 2，合成结果 46.6 度加 5.32； 2 面 180 度加 2，202.5 度加 1.6，225 度加 1.6，合成结果 200.7 度加 4.93							
平衡结果					0.0461	258.5	0.0464	273.2

注：以上实验数据配重单位为 g，幅值单位为 mm，角度单位为度，转速为 1453 转，所加配重半径相同。

MATLAB 程序：

```
A10=0.2186;J10=271.9/180*pi;
A20=0.2057;J20=277.1/180*pi;
A11=0.1175;J11=281.6/180*pi;
A21=0.1158;J21=292/180*pi;
A12=0.1215;J12=278.4/180*pi;
A22=0.1136;J22=290.3/180*pi;
u1=1.2;ju1=90/180*pi;
u2=1.2;ju2=90/180*pi;
X10=A10*cos(J10)+i*A10*sin(J10);
X20=A20*cos(J20)+i*A20*sin(J20);
X11=A11*cos(J11)+i*A11*sin(J11);
X21=A21*cos(J21)+i*A21*sin(J21);
X12=A12*cos(J12)+i*A12*sin(J12);
X22=A22*cos(J22)+i*A22*sin(J22);
U1=u1*cos(ju1)+i*u1*sin(ju1);
U2=u2*cos(ju2)+i*u2*sin(ju2);

y11=(X11-X10)/U1;
y12=(X12-X10)/U2;
y21=(X21-X20)/U1; y22=(X22-X20)/U2;
a=[y11 y12;y21 y22];
c=[-X10;-X20];
d=a^(-1)*c
```

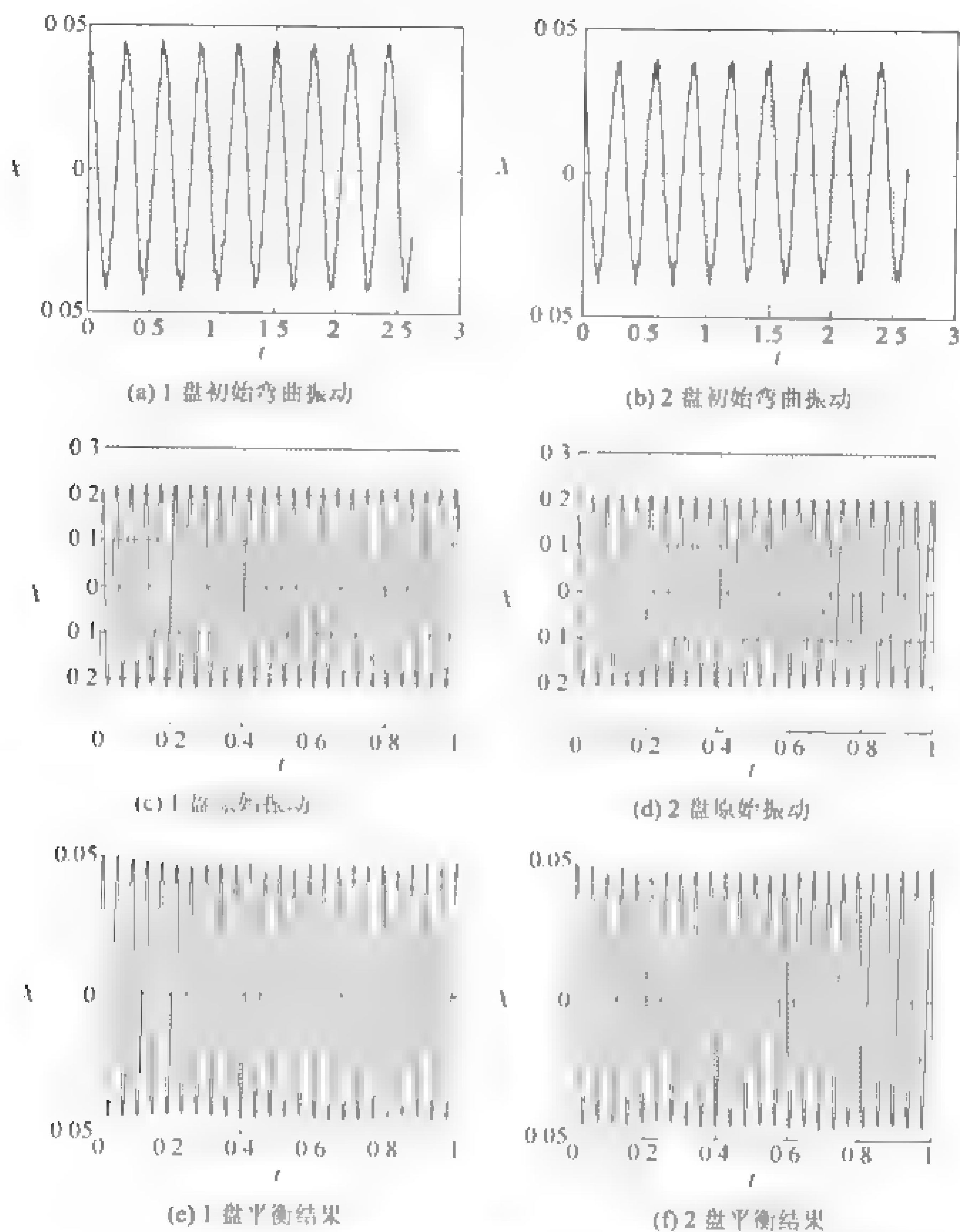


图 8.30 动平衡初始弯曲、原始振动和平衡结果的波形图

附录 A MATLAB 简介

MATLAB 作为一种高效率的工程计算工具，成功地将编程计算和数据图形化集成在另一个易用的环境中，研究人员可以用近似数学表达式的形式来对各类问题进行描述和求解. 因此 MATLAB 在数学、数据采集、建模、仿真、数据分析和图形化等多领域得到了广泛的应用.

由于当前各高校的机械专业和力学专业都在本科教学计划中包含至少一门计算机语言课程，此外考虑到本书主题和篇幅要求，在这里不再对计算机编程的基础思想进行过多介绍，也无法实现细致完备的 MATLAB 大全；而是主要针对 MATLAB 语言和其他高级编程语言的不同之处，对本书中使用到的 MATLAB 功能进行简要的介绍. 力求使掌握计算机编程基础思想的读者能够通过这个快速指南，顺利地理解本书中的 MATLAB 程序.

对于还不了解计算机编程基本思想的读者，我们建议其可以先通过学习《FORTRAN 77 结构化程序设计》(谭浩强，田淑清；清华大学出版社，2000)或者《C 语言程序设计(第 2 版)》(谭浩强；清华大学出版社，2009)等著作来完善这一部分的知识. 在数值仿真日益受到重视的今天，具备计算机编程基本能力，不仅是进行结构动力学有限元分析的前提，而且对进行结构动力学其他研究方向也是有益的.

对于打算进一步深入学习 MATLAB 的读者，我们建议其可以参考 MATLAB 自带的帮助文档和以下专著：《精通 MATLAB 7》(Duane Hanselma；清华大学出版社)或者《MATLAB 实用教程》(徐金明；清华大学出版社)。

A.1 MATLAB 使用界面

MATLAB 7.4 的默认界面如图 A.1 所示.

区域①中显示的是**当前目录**中的文件. 当前目录是指执行函数或者读取数据时，如果对相关文件没有指定绝对路径，则除了系统默认目录以外，系统寻找相关文件的目录. 当前目录可以通过工具栏上的按钮来设置. 系统会显示最近使用过的 20 个目录在按钮右边的下拉式菜单中，对于这些目录，也可以使用该菜单快速指定为当前目录. 如果通过 File→Set Path 命令将某个目录加入了系统默认目录行列，则需要运行或者读取该目录中的文件时，可以不需要事先将该目录设置成当前目录. 通过单击区域①下部的标签，也可以将这一部分切换为显示**工作空间**内容. 工作空间中显示当前创建的各种变量的情况，包括其内容，内存使用大小等.

区域②是命令记录窗口. 在该窗口中记录了最近使用的命令. 对区域①和区域②中显示的内容，可以按照大家所熟悉的 Windows 资源管理器的习惯进行操作. 比如在文件名上双击表示打开该文件，而右击则弹出相应菜单.

区域③是命令窗口. MATLAB 属于解释型语言. 可以不用编写一个程序，而直接通过

在命令窗口中输入代码并运行. 例如在命令窗口可以直接输入如下代码:

```
rho = (1+sqrt(5))/2
```

按回车键后，MATLAB 将会在命令窗口中输出结果(在后面部分将直接用斜体来表示系统输出，而不再专门另行提示):

```
rho =
   1.6180
```

同样的例子如果通过 FORTRAN 语言或者 C 语言来完成，就需要多个步骤：编写，编译和运行完整的程序；完整的程序中又要包括变量定义，算法实现和结果输出等部分. 而除了自定义函数以外的少数情况，大多数 MATLAB 程序中的语句和函数都可以在 MATLAB 命令窗口中输入执行，因此在很多情况下不用编程也可以充分快速地利用 MATLAB 的强大功能. 相比较于要求严格的 FORTRAN 语言或者 C 语言，MATLAB 的这种灵活特性对科研工作来说是有益的：无论是前期的数据预处理，还是程序运算完成后结果的再分析，都可以方便快速地进行.

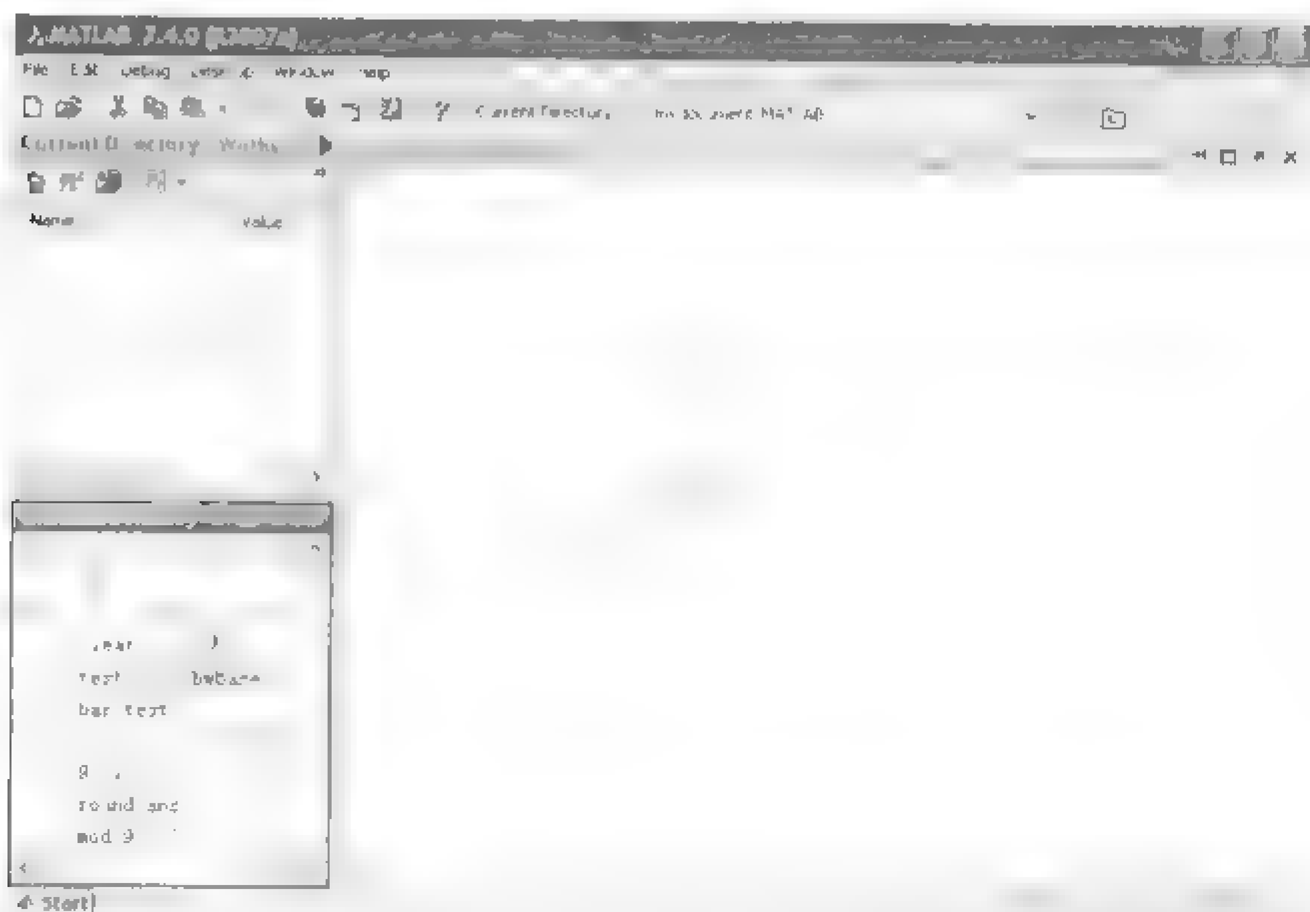


图 A.1 MATLAB 界面

A.2 MATLAB 编程简介

A.2.1 命令文件和函数文件

除了在命令窗口下输入命令来完成操作之外，为完成复杂计算，支持程序概念显然是必须的. MATLAB 支持运行后缀为 M 的程序文件. 考虑到熟悉 FORTRAN 语言或者 C 语

言的读者的习惯，本书将变量等概念移到 M 文件编程部分介绍；不过这不表示这些概念不适用于命令窗口输入执行方式.

M 文件可以分为两类：命令文件和函数文件.

为了代替在 MATLAB 提示符下输入 MATLAB 命令的语句，可以把这些命令写入一个文本文件. 每当用户输入这个文件名时，这些命令就由 MATLAB 执行. MATLAB 从文件而不是从终端读取命令，当文件中最后一个命令被执行时，MATLAB 能再从终端读取命令. 这类文件称为命令文件.

函数文件用来创建函数，也是由 MATLAB 命令构成的，只是在其文件头部有一定的格式要求. 这部分内容将在 A.2.6 节中详细介绍.

函数文件和命令文件的执行如同普通的 MATLAB 命令，不需要进行编译. 当输入文件名时(如果有自变量就一起附带参数)，就执行文件中的语句. 所有的 M 文件都是普通的 ASCII 文本文件，可以使用 MATLAB 自带的编辑器来创建和编辑，也可以使用其他通用文本编辑器来创建和编辑. 在 M 文件中，以%开始的行为注释行. 输入 help name，将显示该 M 文件中的第一个命令之前的所有注释行.

A.2.2 变量

(1) 变量名：MATLAB 变量名可以由字母、数字和下划线组成，但是变量名称必须由字母开始. MATLAB 是字母大小写敏感的. A 和 a 将被视为不同的变量. 变量名长度无限制，但只有变量名的前 *N* 位字符是有效的. *N* 视系统而定，可以用 namelengthmax 函数来确定，一般 PC 上为 63. 可以用 iskeyword 来查看系统保留关键词，除此之外系统内置函数名可以用作变量名. 但是应当注意避免变量名与已有函数重名，否则在程序中将不能调用该重名函数.

(2) 变量类型：MATLAB 支持的变量类型相当广泛. 不但包括各种编程语言中常见的整数型、实数型、字符型和逻辑型等基本类型变量，还支持元胞和构架等类型. MATLAB 中数值型变量直接支持复数，无须专门定义复数型变量. 此外，对矩阵提供了最底层的支持；各变量类型都可建立相应的矩阵，并被视为一个变量，可以直接对其进行运算操作.

(3) 变量定义： MATLAB 内建类型变量并不要求预先定义，可以直接通过变量赋值来建立变量并由其所赋予的数据类型来决定变量的类型. 个别扩展功能组件引入的变量类型需要事先定义. 比如符号运算中的符号变量.

(4) 变量可变性：首先是类型可变性. 大多数计算机语言不允许赋予一个已存在变量与其原本类型不符合的数据，而 MATLAB 中将直接更改变量类型存入新的数据. 其次是大小可变性. 矩阵的大小将由数据来决定，可以在建立变量后再动态改变其矩阵维数. 这些特性是实现 MATLAB 的灵活性所需要的，但是我们建议读者在编写 MATLAB 程序的时候，最好在程序开始时通过赋值来统一建立变量并同时用注释说明变量意义，尽量不要在程序中改变变量意义和类型.

(5) 数组：一般计算机语言中都将数组视为变量的组合. 但 MATLAB 中如前所述，将矩阵作为一种基本数据格式；而实际上数组就是一种矩阵. 所以在 MATLAB 中没有与一

般变量相并列的数组概念. 读者可以从后面的例 9.1 和例 9.2 中体会关于数组(矩阵)的一些具体操作.

A.2.3 算术运算符和算术表达式

算术运算符介绍如下.

(1) “+”, “-”: 分别为加减法运算符, 也可以用来表示数值正负.

(2) “*” : 乘法运算符.

(3) “/” : 除法运算符(矩阵右除).

(4) “\” : 矩阵左除运算符.

(5) “^”: 乘幂运算符.

“*”、“/”、“^”和“\”都是对矩阵整体进行运算, 如果在其前面加上“.”, 则表示对矩阵中的每一个元素单独进行该运算.

在上述运算符中, 靠后的运算级别高. 在带相同优先级的运算符表达式中, 按从左到右的顺序执行. 圆括号( )能够用于改变优先级次序.

使用“'”可以得到复数的共轭. 对矩阵来说 A'得到的是 A 的共轭转置矩阵, A.'是其转置; 对实数矩阵来说两者是一样的.

在 MATLAB 中科学记数法格式如 1.23 E-6(即 1.23×10^{-6}).

【例 A.1】 在 MATLAB 中创建两个矩阵 A 和 B, 并计算 B*A、B*A*B'、A 中第 2 行第 3 列的元素与 B 的乘积、A 中第 1 列元素与 B 的乘积:

```
A=[1 2 3; 2 1 3; 3 2 1]
A =
     1     2     3
     2     1     3
     3     2     1
B=[1 1 1]
B =
     1     1     1
```

建立一个矩阵可以有多种方式, 一般是是在方括号[ ]内逐行给定元素, 行内元素间以空格分开, 行与行间用分号隔开. 作为特例, 如果定义一个标量, 则可不需要方括号.

```
B*A
ans =
     6     5     7
```

在计算 B*A 时, 没有指定表达式的结果应赋予哪个变量, 系统默认在此情况下创建变量 ans 来保存结果. 如果 ans 已经存在, 则以前的值会被冲掉.

```
A=B*A*B'
A =

    18
```

在计算 B*A*B′时，指定将结果赋予变量 A，因此在命令执行后变量 A 中以前的内容被新的值冲掉，由 3×3 的矩阵变成了一个标量.

```
C=A(2,3)*B
C =
     3     3     3
D= A(:,1)*B
D =
     1     1     1
     2     2     2
     3     3     3
```

引用矩阵 A 中第 m 行第 n 列的元素，可以通过 A(m,n)的方式来实现. 因为 MATLAB 中无指针概念，因此不必像 C 语言中那样去深究矩阵(数组)在内存中的行列排列顺序.

引用矩阵 A 中第 m 行的全部元素，可以通过 A(m,:) 的方式来实现. 类似地，矩阵 A 中第 n 列的全部元素，可以通过 A(:,n) 的方式来引用. “:”一般用来建立一个序列. 其格式为：“序列起始值:序列间隔值:序列中止值”. 如果不指定序列间隔值，则默认为 1. 如执行 t=1:2:5，等同于 t=[1 3 5]; 而 t=1:5，等同于 t=[1 2 3 4 5]. 在引用矩阵元素的时候，若只有一个冒号，而没有指定序列起始值和中止值，则默认为从 1 到该矩阵在这个指标上的最大值.

【例 A.2】 建立复数矩阵 A 和 B，分别计算 B'和 B.'.

```
A=[ 1+4i 2+3i; 3+2i 4+i];
B=eye(2)+magic(2)*j
B =
   1.0000 + 1.0000i        0 + 3.0000i
        0 + 4.0000i   1.0000 + 2.0000i
```

i 和 j 是 MATLAB 中的虚数单位. 可以通过 1+4*i 的方式来表达复数，也可以按照数学中的习惯省略乘号直接写成 1+4i. 如果在程序中将 i 和 j 用作了其他变量，就无法通过上述方式来表达复数. 虽然可以通过类似 ii=sqrt(-1)的方法来重新定义虚数单位常量，但是我们建议一般在程序中还是不要再使用 i 和 j 作为其他变量.

在建立矩阵 A 的命令后有一个分号，其作用是在屏幕上不输出此命令的结果.

建立矩阵 B 的命令中使用了虚数单位 j，从结果中可以看出效果和 i 是一样的.

```
B'
ans =
   1.0000 - 1.0000i        0 - 4.0000i
        0 - 3.0000i   1.0000 - 2.0000i
B.'
ans =
   1.0000 + 1.0000i        0 + 4.0000i

        0 + 3.0000i   1.0000 + 2.0000i
```

A.2.4 关系运算符和逻辑运算符

1. 关系运算符

(1) “< ” 小于.

(2) “<=” 小于等于.

(3) “> ” 大于.

(4) “>=” 大于等于.

(5) “==” 等于.

(6) “~=” 不等于.

作为关系表达式的结果，MATLAB 中有独立的逻辑型变量. 可以通过以下方式建立一个逻辑型变量.

```
x = [true, true, false, true, false];
```

逻辑型变量实际上是将 true 记作 1，false 记作 0，并可参与数值计算.

【例 A.3】挑选出矩阵 A 中大于 5 的数值.

```
A = magic(4)
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
T=A>5
T =
     1     0     0     1
     0     1     1     1
     1     1     1     1
     0     1     1     0
A.*T
ans =
    16     0     0    13
     0    11    10     8
     9     7     6    12
     0    14    15     0
```

2. 逻辑运算符

(1) “&” 逻辑与. 返回一个与 A 和 B 相同维数的矩阵. 在这个矩阵中， A 和 B 对应元素都为非零时，则对应项为 1；有一个为零的项则为 0.

(2) “|” 逻辑或. 返回一个与 A 和 B 相同维数的矩阵. 在这个矩阵中， A 和 B 对应元素只要有一个为非零，则对应项为 1；两个矩阵均为零时，则为 0.

(3) “~” 逻辑非. 返回一个与 A 和 B 相同维数的矩阵. 在这个矩阵中， A 是零时，

则对应项为 1；A 是非零时，则对应项为 0.

(4) “xor” 逻辑异或. 返回一个与 A 和 B 相同维数的矩阵. 在这个矩阵中，如果 A 和 B 均为零或均为非零，则对应项为 0；如果 A 或 B 是非零但不是两者同时为非零，则对应项为 1.

【例 A.4】

```
A = [0 1 1 0 1];
B = [1 1 0 0 1] ;
A & B
ans =
 0 1 0 0 1
A | B
ans =
   1  1  1  0  1
~A
ans =
      1  0  0  1  0
xor(A,B)
ans =
     1  0  1  0  0
```

A.2.5　程序流程控制

1. 选择结构

1) if 语句

```
if logical_expression (逻辑表达式)
   statements (程序语句) % Executes if the result of expression is true.
End
```

注：矩阵元素参与的逻辑表达式，当其结果中所有元素为非零值时，才按为 true 对待.

if, else 的用法：

```
if logical_expression
   statements1  % Executes if the result of expression is true.
else
   statements2  % Executes if the result of expression is false.
End
```

elseif 的用法：

```
if expression1
   statements1  % Executes if the result of expression1 is true.
elseif expression2
   statements2  % Executes if the result of expression2 is true.
end
```

2) switch 语句

```
switch expression (scalar or string)
   case value1
      statements      % Executes if expression is value1
   case value2
      statements      % Executes if expression is value2
:
   otherwise
      statements      % Executes if expression does not match any case
end
```

2. **循环结构**

1) for 语句

```
for index = start:increment:end
    statements
end
```

index、start、increment 和 end 分别是循环变量名、循环的初始值、步长和终值. 如果已经有一个向量 x，也可以写成 index=x.

2) while 语句

```
while expression
    statements  % Executes while expression is true
end
```

用来略过循环中指定部分的 continue 语句和跳出循环的 break 语句，因为在本书例子中没有用到过，且也不提倡在规范化的程序中使用，所以不作介绍.

A.2.6 函数

如果将命令文件理解为 FORTRAN 语言中的程序，则 MATLAB 中的函数文件与 FORTRAN 语言的函数和子程序非常相似. 函数文件有如下特征：

(1) 函数文件的第一行必须包含关键字 function，命令文件没有这种要求. 因此，没有这样第一行的 M 文件是命令文件.

(2) 第一行必须指定函数名、输入变量(参数)和输出变量(参数). 输入参数是从 MATLAB 的工作空间复制到函数工作空间的变量.

(3) function [output1, output2,⋯] = function_name(input1, input2,⋯).

(4) 一个函数可以有零个、一个或几个输入参数和返回值.

若只有一个返回值，用来包含返回值的方括号可以省略：

```
function output= function_name (input1, input2, ...)
```

若没有返回值，则必须写方括号：

```
function []=function_name (input1, input2, ...)
```

若没有输入参数，圆括号可省略：

```
function [output1, output2, ...] = function_name
```

若只有一个输入参数，用来包含输入参数的圆括号不可省略：

```
function [output1, output2, ...] = function_name(input)
```

调用函数时的问题，函数返回值变量的问题．必须数目足够．

建议函数取名和文件名一样．调用时所用的变量并不需要与函数文件中定义的变量有相同的名字．

A.3 稀疏矩阵和符号变量及其运算

稀疏矩阵和符号变量作为 MATLAB 中变量的一种，遵循 MATLAB 中对变量的大多数要求．只是这两种变量对于只熟悉 C 语言或者 FORTRAN 语言，而从未接触过 MATLAB 的读者来说是陌生的，而且在本书例子中也使用了上述两种变量，所以单列一节对其进行介绍．

A.3.1 稀疏矩阵

在有限元问题中，结构的刚度矩阵和质量矩阵包含有大量的零元素，即具有稀疏的特点．对于自由度较多的结构，这一特点更加明显．因此直接使用普通的二维数组来记录稀疏矩阵中的全部元素，显然是不经济的，也是不必要的．如果通过适当的编程，仅仅记录稀疏矩阵中的非零元素，就可以大大减少所需的储存空间．并且由于在这种记录方式中非零元素的位置是已知的，所以无须再做判断，就能区分矩阵运算中有零元素参与的部分并将其略去，而仅计算全部参与元素都非零的部分，这样能有效节省计算时间．

但无论是通过仅储存非零元素来节省存储空间，还是通过略过矩阵运算中有零元素参与的部分来加速计算，都需要编写额外的程序来实现．使用 MATLAB 则可以方便地实现上述稀疏矩阵两个优点而不带来额外编程工作量．

1. 创建稀疏矩阵

当通过例 A.1 中的常规方式创建一个矩阵的时候，MATLAB 将默认使用普通矩阵格式(可以和稀疏矩阵相对称为满矩阵)，所以在创建稀疏矩阵的时候，需要特别指出．

有两种方法可以用来创建稀疏矩阵：

(1) 从已有矩阵转换而得．使用 B=sparse(A)，可以将一个已存在的满矩阵 A 转换成稀疏矩阵 B．

(2) 使用相应函数创建一个稀疏矩阵．如 sparse(m,n)，将生成一个 m×n 的所有元素都是 0 的稀疏矩阵；speye(m,n)将生成一个 m×n 单位稀疏矩阵；sprand(A) 将生成与 A 有相同结构的随机稀疏矩阵，且元素服从均匀分布．类似上述功能的函数还有 sprandn，

sprandsym 等；且各函数都有多种调用方式，如 sparse 函数还可以通过使用 5 个参数：sparse(i,j,s,m,n,)生成一个 m×n 的稀疏矩阵，向量 i,j 分别是向量 s 中非零元素在矩阵中的行数和列数. 在这里限于篇幅不能一一介绍，请读者参考 MATLAB 帮助来了解各函数的详细用法.

2. 使用稀疏矩阵

在得到了稀疏矩阵之后，在矩阵的元素读取和修改上，其用法与满矩阵相同. 有以下几点需要注意.

(1) 在稀疏矩阵和满矩阵的混合运算中，结果将以满矩阵形式保存.

(2) 各系统函数在使用稀疏矩阵作为输入参数时，如果输出参数为一个标量或者一个给定大小的向量，输出参数是满矩阵格式，如命令 size；但如果输出参数是一个矩阵，一般就会与输入参数相同为稀疏矩阵格式，如命令 diag.

(3) 少数函数只支持满矩阵格式或者稀疏矩阵格式. 如求解特征值问题的函数 eig 要求其输入参数必须为满矩阵格式；对稀疏矩阵的特征值问题，需要使用相应 eigs 函数. 当使用 eig 函数求解稀疏矩阵的特征值问题时，系统会提示应该使用 eigs 函数.

【例 A.5】创建满矩阵 A，稀疏矩阵 B 和 C，计算 A*B.

```
A = [ 0   0   0   5
      0   2   0   0
      1   3   0   0
      0   0   4   0]
A =
    0     0     0     5
     0     2     0     0
     1     3     0     0
     0     0     4     0
B=sparse(A)
B =
   (3,1)        1
   (2,2)        2
   (3,2)        3
   (4,3)        4
   (1,4)        5
```

下面再通过指定非零元素位置和数值的方法生成和 B 同样的稀疏矩阵.

```
i = [3 2 3 4 1]';
j = [1 2 2 3 4]';
s = [1 2 3 4 5]';
C = sparse(i,j,s,4,4)
C =
   (3,1)        1
   (2,2)        2
   (3,2)        3
```

```
  (4,3)        4
  (1,4)        5
```

注意 i,j 和 s 要求为向量(n×1 矩阵，列向量).

```
D A*B
D =
     0     0    20     0
     0     4     0     0
     0     6     0     5
     4    12     0     0
```

满矩阵 A 和稀疏矩阵 B 的乘积结果 D 为满矩阵.

```
E = full(C)
E =
     0     0     0     5
     0     2     0     0
     1     3     0     0
     0     0     4     0
```

通过 full 函数可以将稀疏矩阵转换为满矩阵.

A.3.2　符号变量和符号运算

在一般计算机语言中，为求解某一表达式的结果，要求表达式中的参与量必须都是数值，其计算结果也以数值方式给出. 例如有两个变量 x 和 y，需要计算(x+1)*(y+2)时，必须要先给出 x 和 y 的数值才可以完成计算. 而在 MATLAB 中可以使用符号运算，直接求解得到上述问题的解析解：2*x+y+x*y+3.

使用符号运算一般是出于两种需要：

(1) 希望得到问题的表达式解. 例如求解某一个表达式的积分或者微分.

(2) 希望得到无数值计算误差的解. 如计算 1/3，无论数值计算有些数值位数有多高，都是有误差的，在后继运算中，这种误差可能会累计起来而越变越大. 而采用符号预算，计算结果用分数表示，就无此问题.

符号运算的缺点是运算速度很慢，不适合计算量很大的工作. 如求解一个 40 阶矩阵的逆，如果是要求数值解，计算量对于主流的个人计算机来说是较小的；但是要求一个同阶符号矩阵的逆，就需要超出一个数量级的计算时间.

MATLAB 中的符号运算功能相当强大，这里简要介绍其基本功能.

1. 创建符号变量

要进行符号运算，首先需要创建符号变量. 可以使用 sym 和 syms 来创建符号变量. 符号变量有两种，一种其内容是符号形式的数值，另一种其内容是符号. 通过 s=sym(a)的形式，可以将 a 的内容转化为符号变量 s. 如果 a 是一个数值，那在 s 中将储存相应数值的符号表达方式. 如果 a 是一个字符串，则创建了以此字符串的符号变量，在这种情况下

最好使用和符号同样的字符作变量名.

【例 A.6】 创建符号变量 s，x；并计算 s*x 和 s/3.

```
a=[1 2; 3 4];
s=sym(a)
s =
[ 1, 2]
[ 3, 4]
```

注意符号矩阵在输出上和数值矩阵的不同.

```
x=sym('x')
x =
x
c=s*x
c =
[   x, 2*x]
[ 3*x, 4*x]
d=s/3
d =

[ 1/3, 2/3]
[   1, 4/3]
```

当需要创建多个类似 x,y 的符号变量的时候，使用 sym 函数的方式便不够简便. 例如需要创建符号变量 x 和 y，需要使用两次 sym 函数. 实际上可以通过 syms 命令来更加方便地完成这一工作. 通过如下方式：

```
syms x y
```

可以一次就得到符号变量 x 和 y. 但是要将一个数值转化为符号表达方式时，必须使用 sym 函数.

2. 代换符号表达式中符号变量

在使用符号运算完成公式推导后，有时需要将符号表达式中的部分或者全部符号代换为数值量；有时需要将其中的某一个符号换成另外一个符号；这两种情况下都可使用 subs 函数.

【例 A.7】 subs 函数的用法.

```
syms a b
s=cos(a)+sin(b);
subs(s,{a,b},[sym('alpha'),2])
ans =
cos(alpha)+sin(2)
```

注意，s 中的符号表达式还是没有变化的. 如果要将新的表达式保存在其他标量中，如 s1，可以将上面函数写成 s1=subs(s,{a,b},[sym('alpha'),2]).

```
subs(s,{a,b},[1,2])
ans =
    1.4496
```

此时由于 s 中所有符号变量已经都被替换为数值，所以所得结果为数值.

3. 其他相关函数

MATLAB 中的符号运算功能十分强大，拥有众多函数. 可以方便地完成求解线形方程组、微分方程、求导和积分等功能. 读者可以参阅 MATLAB 的帮助文档详细研究以上各领域函数. 从下例中可以看出 MATLAB 中符号运算功能的完善.

【例 A.8】 求解微分方程并绘出其解在$[-2\pi,3\pi]$的图形. 微分方程为$y''+y=2\cos(t)$，其边界条件为：$y(0)=0,y(1)=\sin(1)$.

首先通过 dsolve 函数来求解微分方程：

```
s=dsolve('D2y+y=2*cos(t)','y(0)=0','y(1)=sin(1)')
s =
sin(t)*t
```

在 dsolve 函数中默认自变量是 t，大写 D 表示求导，而 D2 表示二级求导.

现在为观察 s 的变化规律，需要绘制其图像. 但不必对其中的自变量 t 在其变化区间内进行多次代换来得到 s 的变化规律，而可以使用 ezplot 函数.

```
ezplot(s,[-2*pi, 3*pi])
```

所绘的图形如图 A.2 所示.

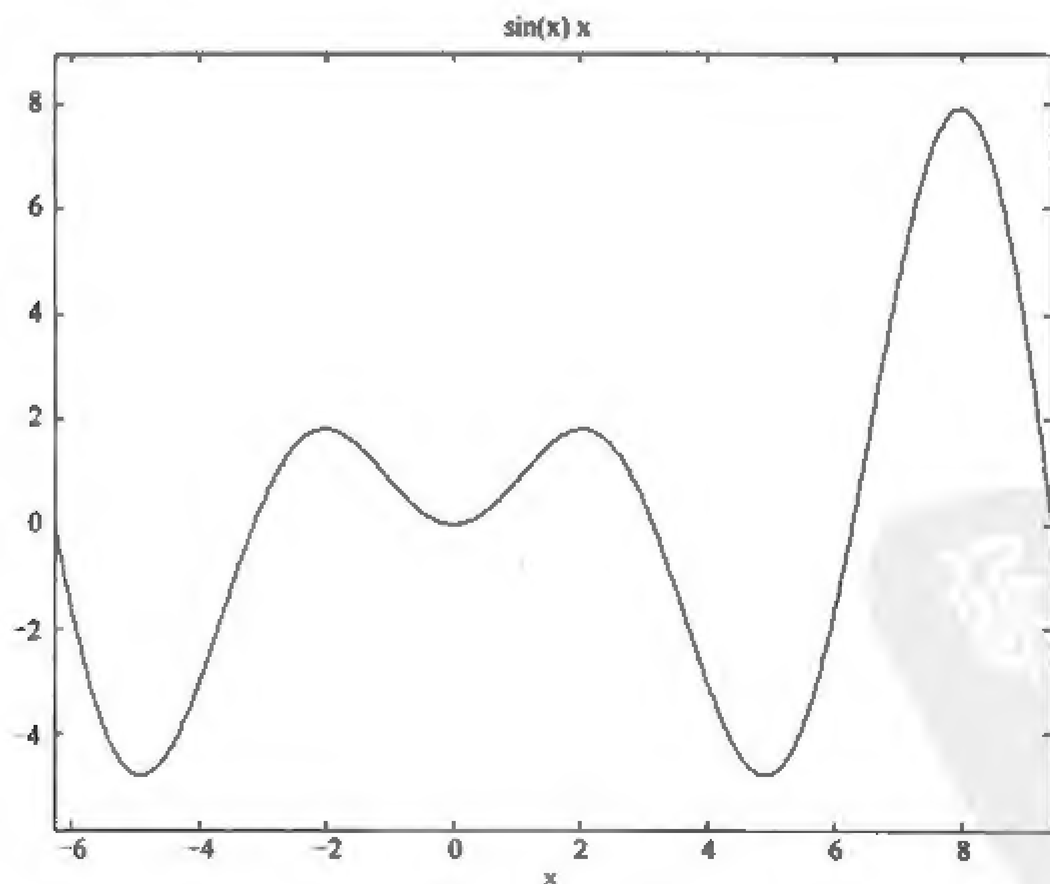


图 A.2　例 A.8 所绘的图形

参 考 文 献

[1] 王勖成，邵敏. 有限单元法基本原理和数值方法[M]. 北京：清华大学出版社，1997

[2] 朱伯芳. 有限单元法原理与应用[M]. 北京：中国水利水电出版社，1998

[3] H.卡德斯图赛；诸德超，傅子智等译. 有限元法手册[M]. 北京：科学出版社，1996

[4] Finlayson BA. The Method of Weighted Residuals and Variational Principles[M]., New York: Academic Press 1972

[5] 张景绘. 动力学系统建模[M]. 北京：国防工业出版社，2000

[6] 薛定宇. 反馈控制系统设计与分析[M]. 北京：清华大学出版社，2001

[7] 傅志方，华宏星. 模态分析理论与应用[M]. 上海：上海交通大学出版社，2000

[8] C. T. F. 鲁斯. 结构力学的有限元法(Finite element methods in structure mechanics)[M]. 北京：人民交通出版社，1991

[9] 匡文起，张玉良，辛克贵. 结构矩阵分析和程序设计[M]. 北京：高等教育出版社，1991

[10] Young W Kwon, Hyochoong Bang. The finite element method using MATLAB[M]. London: CRC Press, 1996

[11] 王焕定，焦兆平. 有限单元法基础[M]. 北京：高等教育出版社，2002

[12] 陈建军，车建文，崔明涛. 结构动力优化设计述评与展望. 力学进展[J]，2001，31(2)：181~192

[13] 顾松年，徐斌，荣见华等. 结构动力学设计优化方法的新进展[J]. 机械强度，2005，27(2)：156~162

[14] 徐斌. 桁架结构动力学拓扑优化和智能桁架结构/控制一体化优化研究[D]. 西安：西北工业大学，2005

[15] Tong WH, Jiang JS, Liu G R.. Solution existence of the optimization problem of truss structures with frequency constraints[J]. International Journal of Solids and Structures, 2000，37：4043~4060

[16] 欧进萍. 结构振动控制——主动、半主动和智能控制[M]. 北京：科学出版社，2003

[17] Xu Bin. Jiang J. S. Integrated optimization of structure and control for piezoelectric intelligent trusses with uncertain placement of actuators and sensors[J]. Computational Mechanics，2004，33(5)：406~412

[18] Xu Bin, Jiang Jiesheng, Tong Weihua, et al. Topology group concept for truss topology optimization with frequency constraints[J]. Journal of Sound and Vibration, 2003，261：911~925

[19] 龚靖，张景绘，徐斌等. 主动桁架模态的试验研究[J]. 振动与冲击，2007，26(6)：47~51

[20] 徐斌，管欣，荣见华. 谐和激励下的连续体结构拓扑优化[N]. 西北工业大学学报，2004，22(3)：313~316

[21] 姜节胜，高跃飞，顾松年. 环境振动试验技术的若干新进展[J]. 机械强度，2005，27(3)：307~311

[22] 顾松年. 结构动力修改的发展与现状[J]. 机械强度，1991，13(1)：1~8

[23] 高跃飞. 结构动力学边界条件优化设计与工程实现方法研究[D]. 西安：西北工业大学，2006

[24] Liu Jike, Yang Qiuwei. A new structural damage identification method[J]. Journal of Sound and Vibration, 2006， 297：694~703

[25] 顾家柳等. 转子动力学[M]. 北京：国防工业出版社，1985

[26] 徐金锁. 带细长轴的动力涡轮转子动力特性分析[D]. 西安：西北工业大学，2005

[27] Shimiu N, Sogabe K. Evaluation of time integration methods [J]. Proceedings of Asia Vibration Conference'89，Shenzhen, China，1989-11-27~29：750~761

[28] 张家忠. 挤压油膜阻尼器-滑动轴承-转子动力系统的非线性动力特性研究：运动稳定性及分岔[D]. 西安：西安交通大学，1997

[29] Newmark N M. A method of computation for structural dynamics [J]. Journal of Engineering Mechanics Division, 1959, 85(EM-3)：67~94

[30] 杨永锋. 机动飞行下裂纹转子的非线性特性研究[D]. 西安：西北工业大学，2006

[31] 张春雷. 旋转机械状态监测与现场动平衡[D]. 西安：西北工业大学，2002